Modelización e implementación de Data Warehouses

Autor: Alejandro Pérez Ortiz

Fecha de defensa: 2 de Julio de 2019

Directora: Vanessa Paulino

Institución directora: everis

Ponente: Oscar Romero Moral

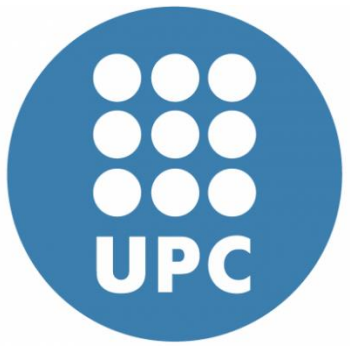
Departamento del ponente: Ingeniería de Servicios y Sistemas de Información (ESSI)

Titulación: Grado en Ingeniería Informática

Especialidad: Ingeniería del Software

Centro: Facultat d'Informàtica de Barcelona (FIB)

Universidad: Universitat Politècnica de Catalunya (UPC) - BarcelonaTech



Agradecimientos

Después de un intenso período de cuatro meses, hoy es el día: escribo este apartado de agradecimientos para finalizar mi trabajo de fin de grado. Ha sido un período de aprendizaje intenso, no solo en el campo conceptual, sino también a nivel personal. Escribir este trabajo ha tenido un gran impacto en mi y es por eso que gustaría agradecer a todas aquellas personas que me han ayudado y apoyado durante este proceso.

Primero de todo, me gustaría agradecer a mi equipo de Data Security en everis por su colaboración. Me habéis apoyado enormemente y siempre habéis estado ahí para ayudarme cuando lo necesitaba. Particularmente me gustaría nombrar a mi compañero en evers, Daniel Macía, con el que he estado trabajando día a día durante mi jornada y me ha enseñado muchísimas cosas, además de ser una grandísima persona.

También me gustaría agradecer a mi ponente, por su valiosa ayuda y aguantar mis múltiples dudas y mis líos mentales. Siempre ha conseguido clarificarme las ideas y me ha ayudado a que el trabajo pudiera llegar a buen puerto.

A mis padres y a mi hermana les tengo que agradecer casi todo, porque han conseguido que yo haya llegado hasta aquí con sus sabios consejos y comprensión.

Por último, me gustaría agradecer a mis amigos de la facultad, una pequeña familia para mí. Adrián, David, Olek, Raúl y Enric, con vosotros he pasado los mejores y los peores momentos de este viaje que ha sido el grado y siempre habéis estado ahí para lo que hiciera falta.

¡Muchas gracias a todos!

Resumen

Partiendo de una arquitectura tradicional de Data Warehouse, se han estudiado y analizado las opciones para transformarla en una estructura Big Data.

Primero se realiza un estudio de las arquitecturas tradicionales de Data Warehouse presentadas por W.H. Inmon y Ralph Kimball viendo que con unos requisitos de gran volumen y alta heterogeneidad de datos no son la mejor opción, con lo que se propone el Big Data como una solución.

Antes de realizar la transformación de la arquitectura, se estudia que es el Big Data y dos de las arquitecturas Big Data más importantes, las arquitecturas Kappa y Lambda, analizando sus componentes y sus ventajas e inconvenientes ya que sobre estas arquitecturas es donde estará la base de la nueva arquitectura. Además, una vez finalizado este estudio, se escogerá una de ellas que será la que más se amolde a la transformación.

Escogida la arquitectura, se mirará por cada uno de los componentes que contiene que tecnologías son las que mejor encajen con los requisitos que se tengan para cada uno de esos componentes y se realizará una comparación entre las diferentes opciones para ver en que es mejor y peor cada una según el caso de uso.

Una vez escogidas las diferentes tecnologías, se verá como conectar cada una de ellas y como se crea una capa OLAP por encima para poder tener un modelado similar al que se tendría en un Data Warehouse tradicional.

Por último, y una vez montada la arquitectura, se propondrá un método para securizar los datos que se almacenen en dicha arquitectura viendo sus diferentes tipos y las diferentes formas en que se puede implementar dentro de una arquitectura.

Resum

Partint d'una arquitectura tradicional de Data Warehouse, s'han estudiat i analitzat les opcions per transformar-la en una estructura Big Data.

Primer es realitza un estudi de les arquitectures tradicionals de Data Warehouse presentades per W.H. Inmon i Ralph Kimball veient que amb uns requisits de gran volum i alta heterogeneïtat de dades no són la millor opció, amb el que es proposa el Big Data com a solució.

Abans de realitzar la transformació de l'arquitectura, s'estudia que és el Big Data i dues de les arquitectures Big Data més importants, les arquitectures Kappa i Lambda, analitzant els seus components i els seus avantatges i inconvenients, ja que sobre aquestes arquitectures és on estarà la base de la nova arquitectura. A més, un cop finalitzat aquest estudi, s'escollirà una d'elles que serà la que més s'adapti a la transformació.

Escollida l'arquitectura, es mirarà per cada un dels components que conté que tecnologies són les que més encaixen amb els requisits que es tinguin per a cada un dels components i es realitzarà una comparació entre les diferents opcions per a veure en què és millor i pitjor cadascuna segons el cas d'ús.

Un cop escollides les tecnologies, es veurà com connectar cadascuna d'elles i com es crea una capa OLAP per sobre per poder tenir un modelatge similar al que es tindria en un Data Warehouse tradicional.

Per últim, i un cop muntada l'arquitectura, es proposarà un mètode per securitzar les dades que s'emmagatzemin en aquesta arquitectura veient els diferents tipus i les diferents formes en què es poden implementar dins d'una arquitectura.

Abstract

Starting from a traditional Data Warehouse architecture, the options to transform it into a Big Data structure have been studied and analysed.

First, a study of the traditional Data Warehouse architectures presented by W.H. Inmon and Ralph Kimball has been done seeing that with high volume and high data heterogeneity requirements, these architectures are not the best option, so Big Data is proposed as a solution.

Before doing the transformation of the architecture, it is studied what is Big Data and two of the most important Big Data architectures, the Kappa and Lambda architectures, analysing their components and their advantages and disadvantages, since on these architectures is where the base of the new architecture will be. In addition, once this study is completed, one of them will be chosen, which will be the one that most adapts to the transformation.

Once the architecture is chosen, it will be looked for each of the components that contain the technologies that best fit the requirements for each of these components and a comparison will be made between the different options to see which technology is better and worse. according to the use case.

Once the different technologies have been chosen, we will see how to connect each of them and how an OLAP layer is created above to be able to have a model similar to the layer that would be in a traditional Data Warehouse.

Finally, once the architecture is assembled, a method will be proposed to secure the data stored in this architecture by looking at its different types and the different ways in which it can be implemented within an architecture.

Contenido

Índice de ilustraciones

Índice de tablas

1 Contexto

1.1 Introducción

Este proyecto se realiza en modalidad B con un convenio de Cooperación Educativa con everis. Dentro de esta empresa, se están realizando actividades dentro de una nueva propuesta en *Data Security* en las cuáles participo y que han motivado a la ejecución de este proyecto.

Para comenzar a adentrarse dentro del terreno del proyecto, hay que hablar de *BI*[1], que es un término que incluye las aplicaciones, infraestructuras y herramientas y las *best practices*[2] que permiten acceder y analizar la información para mejorar y optimizar las decisiones y el rendimiento. Dentro de estas infraestructuras, encontramos el *DW*[3], que es una arquitectura de almacenamiento que guarda los datos extraídos de sistemas transaccionales, almacenes de datos operacionales y fuentes externas . Una alternativa al DW, sería el *Big Data*, el cual se define como activos de información de gran volumen, alta velocidad y/o gran variedad que demandan formas de procesamiento de información innovadoras y rentables que permiten una visión mejorada, toma de decisiones y automatización de procesos, y las tecnologías que lo forman se suelen denominar *NoSQL*, que son una amplia clase de sistemas de gestión de bases de datos que difieren del modelo clásico de SGDBR[4] en aspectos importantes, siendo el más destacado que no siguen el modelo relacional ni la arquitectura de las bases de datos tradicionales.

Las empresas cada vez están haciendo más uso de los datos que tienen, por ello el *Data Warehousing* se ha convertido en una de las tecnologías e iniciativas más importante de una empresa dentro del *BI*.

1.2 Formulación del problema y objetivos

En esta sección se verá dónde surge la idea de este proyecto y cuáles son los objetivos que se quieren cumplir con la realización de este proyecto.

1.2.1 El problema

El mundo de los datos está evolucionando rápidamente y los DW tienen ciertas limitaciones. Actualmente nos encontramos que las empresas están manejando grandes cantidades de datos, que éstos provienen de una gran variedad de fuentes de datos y existen una gran variedad de maneras en la que los datos vienen estructurados. Además, la entrada de estos datos viene dada con frecuencias muy diversas, desde semanales o diarios hasta en tiempo real. En casos con estos requisitos, los DW no son una solución.

[1] *Business Intelligence.*
[2] Conjunto de acciones que han tenido un buen rendimiento o incluso un excelente servicio en un determinado contexto, y que se espera que, en contextos similares, den similares resultados.
[3] *Data Warehouse.*
[4] Sistema de Gestión de Bases de Datos Relacionales.

1.2.2 Objetivos

Los objetivos de este proyecto se podrían ver como una jerarquía en la cual tenemos un objetivo principal, del cual cuelgan otros objetivos necesarios para la correcta ejecución del objetivo principal.

El objetivo principal de este proyecto **es orquestar los elementos que componen las arquitecturas Big Data y las diferentes tecnologías compatibles para poder llevar el concepto de DW a esa arquitectura**.

Para conseguir este objetivo, hay que cumplir con otros que surgen como camino para llegar a realizar el objetivo más importante de este proyecto:

- Estudio de las arquitecturas tradicionales de DW que encontramos en las empresas.
- Ver que limitaciones tienen las arquitecturas tradicionales de DW cuando manejan grandes cantidades de datos, con éstos proviniendo de una gran variedad de fuentes de datos y con una gran variedad de frecuencias en qué éstos llegan al sistema.
- Análisis de las arquitecturas Big Data que se están implementando en las empresas.
- Aplicación de seguridad en la arquitectura Big Data para proteger los datos que contiene.

1.3 Actores implicados

Una vez definido el problema que se quiere resolver y los objetivos que se quieren cumplir, se va a ver a quién va dirigido el proyecto, quien lo usará y quien se beneficiará de su desarrollo.

1.3.1 Desarrollador

Es el encargado de la investigación y la documentación del proyecto. Además, aunque se comunique con el director y el ponente del proyecto, es la última persona en la toma de decisiones y, por tanto, es el encargado de la gestión del proyecto y de cumplir con todos los plazos del proyecto.

1.3.2 everis – Departamento de Data Security

Es el departamento en el cual trabaja el desarrollador. El proyecto será utilizado cómo formación para el departamento acerca de los diferentes tipos de arquitecturas que se pueden encontrar en cliente, ya sean de DW o de Big Data, y así poder plantear soluciones para los clientes con una base sólida. Además, la sección dedicada al análisis de la seguridad de la arquitectura Big Data tendrá más importancia para este departamento ya sea para formación de los integrantes del propio o incluso para incluirlo dentro del proyecto llevado a cabo por este departamento. También darán soporte al desarrollador tanto para corregir errores dentro del proyecto como para resolver dudas.

1.3.3 Director y Ponente del proyecto

Son los principales encargados de guiar y en general, ayudar al desarrollador. Su actuación es indispensable para corregir errores y/o desviaciones en el transcurso del proyecto. En particular, Vanessa Paulino de everis actuará como directora del proyecto y Oscar Romero, profesor de la FIB, actuará como ponente.

1.3.4 Los clientes

Son los que se beneficiarán de este proyecto, ya que como este proyecto servirá de formación al equipo de consultores del departamento de Data Security, éstos podrán ofrecer a los clientes soluciones y/o arquitecturas con una base de conocimiento más sólida y, por tanto, podrán proporcionarles soluciones óptimas y fiables.

2 Estado del arte

En esta sección se va a realizar un estudio de los proyectos que existen actualmente en el mercado sobre el ámbito de este proyecto y ver que limitaciones tienen y como se va a diferenciar este proyecto de ellos.

Existen diferentes estudios sobre las arquitecturas de DW, entre los cuales se encuentran los realizados por Ralph Kimball y W.H Inmon en sus respectivos libros[5], los cuales se centran en explicar las diferentes características que poseen y las ventajas de las cuáles nos vamos a poder aprovechar, pero no hablan acerca de las limitaciones que podemos encontrar dentro de éstas.

También podemos encontrar estudios acerca de las arquitecturas Big Data, con la arquitectura Lambda y la arquitectura Kappa como referentes. Pero en ellos, no podemos encontrar el paso que hay de una arquitectura de DW a una de Big Data.

Con estas dos vertientes, podemos ver que existe un vacío en cuanto al paso de las arquitecturas DW a las arquitecturas Big Data y en qué casos hay que realizarlo.

Viendo estas dos vertientes, podemos ver que se está diferenciando entre dos arquitecturas y, por lo tanto, se están entendiendo como dos conceptos totalmente opuestos. Este hecho, hace que muchas empresas tengan miedo a dar el cambio ya que piensan que tendrán que invertir en una nueva arquitectura sin poder utilizar nada de lo que tenían y, además, tendrán que formar a sus empleados para que aprendan a manejar esta nueva arquitectura.

Con este miedo lo que sucede es que no renuevan sus arquitecturas y, o crean una nueva arquitectura donde añadirán pequeños conjuntos de datos o los nuevos datos que lleguen, o utilizan las arquitecturas que ya tienen para cumplir con los nuevos requisitos del mercado, perdiendo rendimiento.

Aprovechando esta necesidad, surge este proyecto el cuál partirá de una arquitectura Big Data y orquestará los componentes de ésta para llevar el concepto de DW a dicha arquitectura.

Sobre el ámbito del proyecto, existen algunas herramientas aisladas que llevan el concepto de DW al mundo NoSQL, por ejemplo, Apache Druid, pero en general no hay una manera estándar de desarrollar una arquitectura DW con tecnologías NoSQL.

[5] 'The data warehouse toolkit: the definitive guide to dimensional modelling' y 'Building the data warehouse', respectivamente.

3 Alcance y obstáculos

En esta sección se mostrarán que puntos van a formar parte del estudio del proyecto y cuáles son los posibles obstáculos que podrían aparecer durante la ejecución del proyecto.

3.1 Alcance

El desarrollo de este proyecto empezará con un estudio de las arquitecturas tradicionales de DW diseñadas por Ralph Kimball y W.H. Inmon viendo en que componentes están organizadas y que características tienen.

Una vez realizado este estudio, se analizarán las limitaciones que tienen a la hora de abarcar grandes cantidades de datos, con estos viniendo de una gran variedad de fuentes de datos y con una estructuración de datos heterogénea y que sucede si se les ingesta datos en periodos de tiempos muy cortos, como podría ser la ingesta en tiempo real de información de un sensor.

Una vez vistas estas limitaciones, se estudiarán las arquitecturas Lambda y Kappa para sistemas Big Data y se decidirá una de las arquitecturas para el siguiente paso. Una vez seleccionada, se estudiará como orquestar los diferentes componentes de esa arquitectura para simular el comportamiento de un DW dentro de esa arquitectura.

Por último, y con la llegada de las nuevas leyes de protección de datos (como GDPR[6]), se estudiará como securizar toda esta arquitectura Big Data para poder proteger los datos que incluya.

3.2 Obstáculos

3.2.1 Calendario

Una mala planificación de las iteraciones puede hacer que se llegue al final sin tiempo para poder realizar la última iteración. Para evitarlo se intentará hacer una planificación realista y también se utilizará la comunicación semanal para evitar esas desviaciones de tiempo.

3.2.2 Imprevistos que impidan al desarrollador

Imprevistos como ponerse enfermo podrían hacer que no se cumplieran los tiempos marcados en la planificación. Una solución a esto es poner en la planificación unos días de comodín, es decir, unos días sin carga de trabajo que si se pueden utilizar para el proyecto servirían para mejorar algunos puntos y sino, para los imprevistos.

[6] Reglamento General de Protección de Datos

3.2.3 Pérdida de información

Un apagón o problemas con el *hardware* o *software* podrían hacer perder las últimas modificaciones de un fichero. Para intentar evitar este suceso, están los métodos de seguimiento con su control de versiones y sino, se puede utilizar el método de recuperación de la herramienta de redacción que se utilizará, Microsoft Word, aunque no siempre lo solucione. Además, se utilizará almacenamiento en la nube para más seguridad.

3.2.4 Problema proyecto teórico

Es un problema con bastantes probabilidades de surgir en un proyecto teórico si no se hace una buena planificación o si durante el desarrollo no se tiene en cuenta dicha planificación. Este problema consiste en que cuando se está en la fase de desarrollo se encuentre mucha información acerca de algún punto y realizando dicha sección se exceda en información y en tiempo ya que se quiere explicar mucho más y, por lo tanto, afecte a las demás tareas y, por lo tanto, a la fecha de entrega. La solución a esto es centrarse en la planificación cuando se realice una tarea y si se han estimado X horas para ella, por mucha información que haya sólo se dediquen X horas.

3.2.5 Exceso de tecnologías de estudio

Este problema podría surgir a la hora de hacer el mapeo entre las arquitecturas Big Data y las arquitecturas de DW para poder hacer el DW en la arquitectura Big Data. Al ver cuál componente simularía a cuál y que tecnología se utilizaría, podrían salir una gran cantidad de tecnologías que cumplen con los requisitos y, en ese caso, habría que marcar un número límite de tecnologías que se analizarán ya que, si no, no se podría cumplir con los plazos.

4 Metodología y rigor

Una vez visto hasta qué punto de estudio llegará el proyecto y cuáles son los posibles obstáculos que se podrían encontrar durante el desarrollo de éste, esta sección va a mostrar cual va a ser la metodología de trabajo que se va a seguir, qué herramientas se van a utilizar para realizar un seguimiento del proyecto y cómo se va a validar que el proyecto se está realizando como debería.

4.1 Métodos de trabajo

Al ser un proyecto de una duración relativamente corta (aproximadamente 4 meses) lo más apropiado es seguir una metodología ágil ya que proporciona flexibilidad, por si hubiera que cambiar algunos requisitos y/o solución, desarrollo rápido y entregables en menos tiempo que otras metodologías.

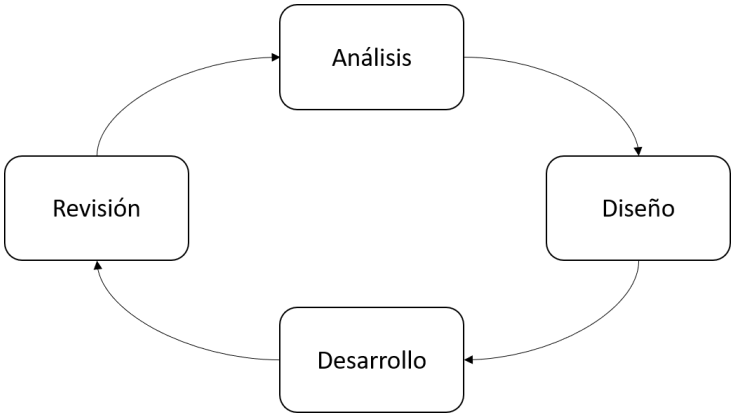


Ilustración 1: Fases de cada tarea

En la podemos ver las diferentes fases por las cuáles pasará cada una de las tareas del proyecto.

El proyecto se dividirá en varias iteraciones que mapearán con los objetivos, y serán:

1. Estudio de las arquitecturas tradicionales de DW

2. Estudio de las limitaciones de estas arquitecturas respecto a los requisitos definidos

3. Estudio de las arquitecturas de Big Data

4. Orquestación de los diferentes componentes de esta arquitectura para simular el DW en ella

5. Securización de la arquitectura Big Data.

Estas tareas, a su vez se dividirán en subtareas más pequeñas para poder ir haciendo entregables más asiduamente.

4.2 Herramientas de seguimiento

Para ver la evolución del proyecto se utilizarán básicamente dos herramientas, Google Drive y Microsoft Teams, en los cuales se hará un sistema de directorios en el cual estará la carpeta raíz con la versión más reciente que se haya entregado/revisado con nombre “Proyecto vX.0” siendo X la versión del documento, y otro documento que será el editable de esa versión con nombre “Proyecto vX.x” siendo x la versión más actualizada dentro de lo que no se ha mostrado/corregido aún. En esta carpeta, además, habrá una carpeta llamada “Old” donde estará todos los ficheros “Proyecto vY.0” siendo Y<X.

Además, se tendrá activada la opción que nos proporciona Microsoft Word de control de cambios para ver qué fue lo último que se editó en la última modificación del documento.

4.3 Métodos de validación

Al final de cada iteración se realizará una reunión tanto con el responsable del proyecto dentro de everis como con el ponente para recibir *feedback*[7] sobre esa iteración y detectar que es lo que se tiene que corregir durante la siguiente iteración.

Durante una iteración, habrá comunicación semanal con el responsable de everis y el ponente para ir corrigiendo desviaciones e ir resolviendo los problemas que puedan surgir durante ésta.

Para estas reuniones y tener un seguimiento más completo, se utilizará Trello para llevar la planificación de tareas al día y poder añadir en cada tarea los recursos empleados.

En estas reuniones, aprovechando la información que se guardará en el Trello, se realizará una comparación entre las horas y costes estimados y las horas y costes reales para ver si son iguales o hay cambios. En este último caso, se intentará ver cuál es la causa de esa desviación y cuál es el plan de acción para corregirla.

[7] Reacción, respuesta u opinión que da un interlocutor como retorno sobre un asunto determinado.

5 Planificación del proyecto

En esta sección se tratará la planificación temporal del proyecto, a la vez que se describirán las diferentes tareas que se van a llevar a cabo dentro de la ejecución del proyecto. Además, se definirán planes de acción para poder corregir desviaciones que puedan surgir en el transcurso de éste.

El proyecto empezó el 19 de febrero de 2019 y tiene como fecha límite el 24 de junio, una semana antes de las fechas de defensa del proyecto. Por ello, la planificación se ha hecho en torno a estas dos fechas.

5.1 Descripción de las tareas

En esta sección se describirán las diferentes tareas que se han planeado para llevar a cabo el proyecto.

5.1.1 Gestión del proyecto

Esta parte será la que se realizará durante las primeras semanas de la vida del proyecto. Se dividirá en dos iteraciones: definición del alcance y gestión del proyecto.

En la iteración 1, se detallarán los objetivos y el alcance del proyecto, los actores implicados dentro del proyecto y se describirá la necesidad por la cual nace este proyecto.

En la iteración 2, se detallarán las diferentes tareas que se realizarán en el transcurso del proyecto, las diferentes dependencias entre ellas, una estimación del tiempo que durará cada una de ellas y el plan de acción y alternativas por si surgieran imprevistos.

Además, se estudiarán los costes del proyecto en cuanto a recursos, ya sean materiales o humanos y la sostenibilidad de éste, centrándose en el ámbito económico, social y ambiental.

Por último, se llevará a cabo la gestión para concretar las diferentes reuniones y controles que se realizarán para asegurar que el proyecto va en buena dirección y para detectar desviaciones y actuar.

5.1.2 Estudio de las arquitecturas tradicionales de DW

En esta tarea se explicará primero el concepto de DW, definiéndolo, y describiendo los objetivos que tiene, es decir, que necesidades cubre. Tras esto, se estudiarán dos arquitecturas de DW, que son las propuestas por Ralph Kimball y W.H Inmon. El estudio de estas arquitecturas se centrará en las características que tienen, los componentes que las forman y cómo están organizados estos para su funcionamiento.

Se ha decidido explicar únicamente estas dos arquitecturas ya que son las más importantes y el tiempo del que se dispone para la realización del proyecto no permite extenderse más en esta parte.

Con esto, podemos diferenciar tres iteraciones dentro de esta tarea: la definición del DW, el estudio de la arquitectura propuesto por Ralph Kimball y el estudio propuesto por W.H. Inmon.

5.1.3 Estudio de las limitaciones de las arquitecturas tradicionales

En esta tarea se tomará como base el estudio realizado en la anterior tarea. Con esta información, se detectarán que limitaciones tienen las estructuras tradicionales a la hora de trabajar con casos con grandes cantidades de datos, con éstos proviniendo de una gran variedad de fuentes de datos, con una gran variedad de maneras en que éstos vienen estructurados y, además, con una velocidad de entrada de éstos muy diversa, desde semanalmente hasta en tiempo real. Una vez detectadas dichas limitaciones, se verá si hay algo en común entre ellas para encontrar la causa.

Esta tarea solo constará de una iteración ya que se centrará en hacer el estudio de que sucede al someter esas arquitecturas a esos requisitos y sacar conclusiones de los resultados.

5.1.4 Estudio de las arquitecturas Big Data

En esta tarea, se presentará el Big Data como una arquitectura que puede solventar los requisitos mencionados anteriormente. Para ello, se hará un estudio de las arquitecturas más populares dentro de estos ecosistemas, las arquitecturas Lambda y Kappa, viendo sus principales características, cuáles son sus componentes y la organización de éstos.

Como sucedía en la tarea del estudio de las arquitecturas de DW, en esta tarea se ha decidido únicamente estudiar las arquitecturas Lambda y Kappa debido a que son las más importantes y que el tiempo del proyecto es limitado.

Esta tarea se dividirá en tres iteraciones: definición de Big Data, estudio de la arquitectura Lambda y estudio de la arquitectura Kappa

5.1.5 Orquestación de las arquitecturas Big Data

Una vez realizado el estudio anterior, en esta tarea se decidirá cuál de las arquitecturas, Lambda o Kappa, se utilizará para continuar con el proyecto. Una vez decidida, se hará un estudio más exhaustivo de cada uno de los componentes que la forman y se estudiarán que tecnologías se pueden implementar en cada uno de los componentes para lograr simular un DW, estudiando su nivel de madurez, compatibilidad con otras tecnologías, … Por último, se realizará una comparación para ver cuándo se debe utilizar cada tecnología.

Esta tarea también se dividirá en tres iteraciones: estudio de los componentes, estudio de las tecnologías y comparativa de tecnologías.

5.1.6 Securización de las arquitecturas Big Data

Una vez realizada la implementación del ecosistema, se hará un estudio de una de las necesidades más importantes que han surgido en los últimos años como es la protección de los datos. Se estudiarán diferentes alternativas de protección de los datos almacenados en la arquitectura implementada. Una vez realizado el estudio, se estudiarán las diferentes tecnologías que pueden proporcionar dicha seguridad, viendo las principales características que ofrecen y cómo proporcionan esa seguridad.

Esta tarea se dividirá en dos iteraciones: estudio de las técnicas de protección de datos y estudio de las tecnologías.

5.1.7 Revisión del proyecto

Una vez terminadas todas las tareas anteriores, se hará una revisión del trabajo realizado y se verán los recursos que se han utilizado para la realización del proyecto y el tiempo empleado en éste, así como si se ha cumplido con los planes que se habían hecho en la estimación. Esta tarea únicamente tendrá una iteración.

5.2 Definición de dependencias

Una vez descritas las diferentes tareas a realizar en el proyecto, ya se pueden empezar a intuir las dependencias entre éstas. Primero de todo, se realizará la planificación del proyecto ya que, sin ella, no se van a poder realizar las otras tareas. Cuando se empiece con el desarrollo del contenido del proyecto hay dos vías —debido a qué los requisitos están definidos con anterioridad y no extraídos del estudio— el estudio de las arquitecturas tradicionales y el estudio de las arquitecturas Big Data ya que no dependen de otras tareas. En este caso, se empezará por el estudio de las arquitecturas tradicionales ya que las dependencias que surgen después hacen que sea el camino más eficiente.

La tarea del estudio de las limitaciones de las arquitecturas tradicionales depende del estudio de estas arquitecturas, ya que se necesita una base de conocimiento sobre éstas. Una vez realizadas estas dos partes, se estudiarán las arquitecturas Big Data.

La parte de orquestación de arquitecturas Big Data es la que más dependencias tiene ya que necesita una base de conocimiento de las arquitecturas Big Data, pero además necesita conocer acerca de la arquitectura tradicional de un DW para así poder llevar ese concepto a las arquitecturas Big Data.

La tarea de securización de las arquitecturas Big Data, requiere de la orquestación de éstas para saber cómo se han organizado los diferentes componentes y ver cómo se puede incluir la seguridad en ellas.

Por último, la revisión depende de todas las tareas anteriores, ya que se revisará que se ha realizado todo el trabajo marcado y se juntará en un único documento.

5.3 Tiempo estimado

La *Tabla 1* muestra una estimación del número de horas que se dedicarán a cada tarea.

Además, se ha asignado un código a cada tarea para que sea más sencillo referirse a cada una de ellas en el desarrollo del proyecto.

Tarea	Código de tarea	Tiempo estimado (h)	Dependencias
Planificación del proyecto	PPO	80	
Definición del alcance	PPO-A	40	
Gestión del proyecto	PPO-G	40	PPO-A
Estudio de las arquitecturas tradicionales de DW	ADW	54	PPO
Definición de DW	ADW-D	10	
Estudio de la arquitectura Kimball	ADW-K	22	
Estudio de la arquitectura Inmon	ADW-I	22	
Estudio de las limitaciones de las arquitecturas tradicionales	LAT	30	ADW-K, ADW-I
Estudio de las arquitecturas Big Data	ABD	54	
Definición de Big Data	ABD-D	10	
Estudio de la arquitectura Lambda	ABD-L	22	
Estudio de la arquitectura Kappa	ABD-K	22	
Orquestación de las arquitecturas Big Data	OBD	141	ADW, LAT
Estudio de los componentes	OBD-C	54	
Estudio de las tecnologías	OBD-T	54	OBD-C
Comparativa de tecnologías	OBD-TC	33	OBC-T
Securización de las arquitecturas Big Data	SBD	98	OBD
Estudio de las técnicas	SBD-T	44	
Estudio de las tecnologías	SBD-TS	54	SBD-T
Revisión del proyecto	RPO	21	SBD
Total		**478**	

Tabla 1: Tiempo estimado para cada tarea.

5.4 Diagrama de Gantt

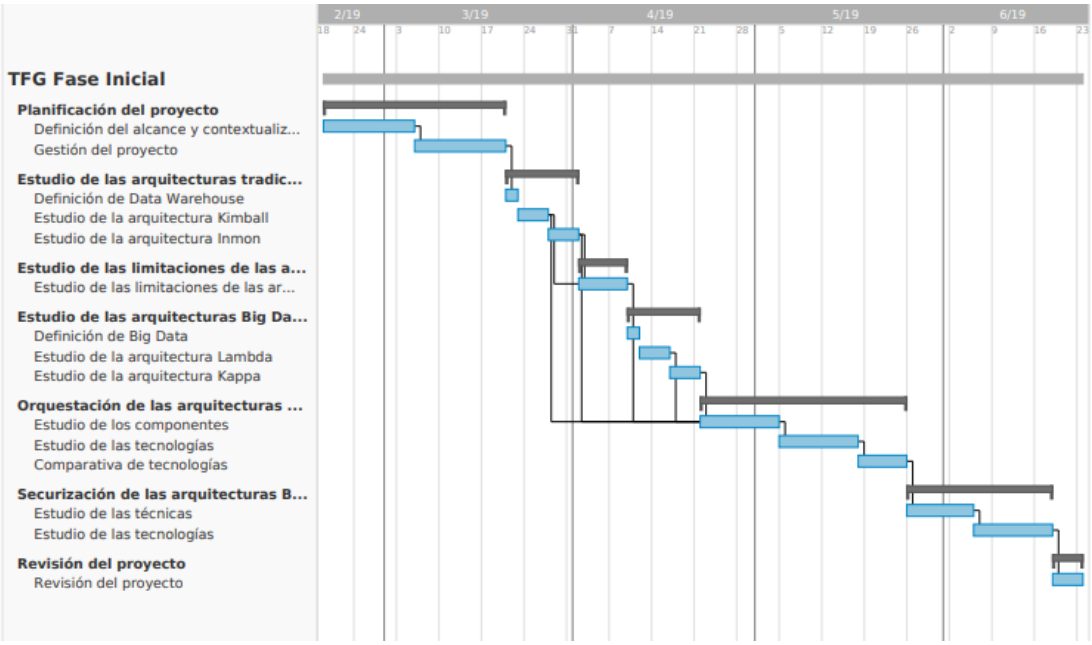


Ilustración 2: Diagrama de Gantt del proyecto
Generado con https://teamgantt.com

5.5 Recursos

En esta sección se definirán los diferentes recursos que se utilizarán durante la realización del proyecto.

5.5.1 Recursos Hardware

Para la realización de este proyecto, al ser un proyecto teórico, únicamente se necesitará un ordenador para el desarrollo del documento y para poder buscar las diferentes fuentes para poder estudiar más acerca del tema. Por lo tanto, se utilizará el ordenador del que dispone la persona que desarrolla el proyecto, es decir, un MacBook Pro.

5.5.2 Recursos Software

En cuanto a los recursos software, se utilizarán dos sistemas operativos distintos de los cuales ya dispone el ordenador que se posee, Windows 10 Home y macOS Sierra. Para el desarrollo de documentos, se utilizará Microsoft Word, para el desarrollo de presentaciones Microsoft PowerPoint y para la gestión del proyecto Microsoft Teams, por lo que se ha optado por obtener una licencia de Microsoft Office que incluye estas herramientas entre otras.

Para la gestión, además de Microsoft Teams, se utilizará Trello, ya que dispone de paneles para poder colocar las tareas y poder acceder a ellas de manera simple y rápida.

Para la realización del Gantt se ha optado por utilizar el software online TeamGantt ya que era gratuito y ofrecía las funcionalidades adecuadas para poder hacer conjuntos de tareas y dependencias.

5.5.3 Recursos humanos

El proyecto será desarrollado por una única persona, por lo tanto, ésta tendrá que asumir dos roles diferentes durante el proyecto: gerente del proyecto y desarrollador. Existirán diferentes reuniones de control, tanto con el ponente como con el director del proyecto. Por lo tanto, aquí aparece el rol del director, el cual hay que tener en cuenta a la hora de realizar los presupuestos.

El rol de gerente del proyecto básicamente tendrá como función planificar el proyecto. El desarrollador será el encargado de trabajar la tarea que toque durante la etapa del proyecto en que esté, formándose sobre el tema en cuestión y desarrollando el documento que se entregará al final de la iteración. El director únicamente dará soporte en las diferentes reuniones de control para ver que el proyecto marcha sobre lo previsto.

5.5.4 Otros recursos

Dentro de esta categoría entrarían aquellos recursos que no se pueden categorizar como ninguno de las clasificaciones anteriores. Así que en esta categoría entrarán aquellos recursos que tienen que ver con el espacio donde se realizará el proyecto. Aquí hay que tener en cuenta tanto la iluminación de la sala como la fibra óptica para poder tener conexión a Internet.

5.6 Plan de acción y alternativas

La idea es trabajar tal y como está planeado, llevando a cabo las tareas durante las fechas que están marcadas en el diagrama de Gantt de la *Ilustración 2.* Con este plan y como podemos ver en la *Tabla 1* se necesitará un tiempo total de 478 horas para el proyecto. Considerando que la planificación del proyecto ocupará aproximadamente 4 semanas, durante estas se trabajará a 20 horas por semana. Una vez finalizada la planificación del proyecto, quedarán 14 semanas para el desarrollo de éste. Por lo tanto, se dedicarán 29 horas semanales para poder llevar a cabo el proyecto.

Ya se sabe que dentro de un proyecto surgen imprevistos y errores en las estimaciones realizadas y, por lo tanto, la duración real de una tarea puede variar de la duración que se había estimado. Un escenario en el que se puede producir esto es que la ejecución de una tarea lleve menos tiempo del estimado y en ese caso, se empezará con la siguiente tarea adelantando todo el calendario, y, además, se realizarán las reuniones pertinentes de fin de tarea.

En el otro extremo, tenemos la situación de que una tarea se alargue más de lo estimado. Para este caso, en la estimación realizada en la *Tabla 1* se han sumado 2 horas semanales extras para posibles desviaciones que puedan surgir. Éstas 2 horas semanales suman un total de 36 horas, con lo cual se dispondrá de un día y medio de margen. Si aun así no fuera posible llegar a los plazos estimados, se utilizaría el tiempo de la tarea RPO para corregir las desviaciones y llegar a los plazos.

Si hubiese aún falta de tiempo se optaría por simplificar la tarea SBD y en vez de realizar un estudio de las técnicas de securización de la arquitectura y después un estudio de las tecnologías se optaría por estudiar únicamente una de las técnicas, probablemente el enmascaramiento y se realizaría el estudio de las tecnologías que hay en el mercado que realicen enmascaramiento.

También podrían surgir problemas en la tarea OBD, ya que debido a la falta de información o de un estándar, se podría encontrar un punto en el cuál hubiera algún retraso de tiempo mientras se encuentra una solución. En este caso, también se podrían utilizar los días de margen.

Sólo en el caso que una cantidad grande de tareas tardasen más de lo estimado en un primer punto, el proyecto no llegaría a estar acabado a tiempo. Pero considerando que se ha hecho una división de tareas suficientemente específica, que se ha realizado una estimación realista del tiempo de cada una de ellas y que además se han tenido en cuenta ciertas desviaciones añadiendo horas extras, no se considera probable que se dé esta situación. Por lo tanto, se puede garantizar que el proyecto se finalizará en el tiempo establecido.

6 Presupuesto y sostenibilidad

Esta sección tratará sobre el presupuesto y la sostenibilidad del proyecto. Para detallar estos dos puntos, contendrá una descripción detallada de los costes del proyecto, separándolos por diferentes ámbitos (hardware, software, …), un análisis de como las desviaciones pueden afectar en este presupuesto y una reflexión acerca de la sostenibilidad del proyecto.

6.1 Presupuesto del proyecto

En esta sección se realizará una estimación del presupuesto necesario para cubrir el proyecto. Se dividirá en 5 secciones: hardware, software, recursos humanos, costes indirectos y costes imprevistos, dependiendo de los recursos que se tengan en cuenta. Para la realización de los presupuestos, se utilizarán los recursos marcados en la sección 5.5. Al final habrá una sección de presupuesto total, sumando los presupuestos de las diferentes secciones y teniendo en cuenta los impuestos.

Para calcular las amortizaciones, se tendrán en cuenta dos factores: la vida útil del recurso y la duración del proyecto que son aproximadamente 4 meses y medio.

Únicamente se calculará el coste por tarea en el presupuesto de recursos humanos ya que los recursos tanto de software como de hardware se utilizarán para todas las tareas.

6.1.1 Presupuesto de hardware

La *Tabla 2* contiene los costes del hardware que se va a utilizar durante el desarrollo del proyecto.

Producto	Precio	Vida útil	Amortización
MacBook Pro	1800 €	4 años	168,75 €
Total	**1800 €**		**168,75 €**

Tabla 2: Presupuesto de hardware

6.1.2 Presupuesto de software

La *Tabla 3* refleja los costes del software que se va a necesitar para la ejecución del proyecto.

Producto	Precio	Vida útil	Amortización
Windows 10 Home	145 €	3 años	18,13 €
macOS Sierra	21,99 €	3 años	2,75 €
Microsoft Office	69 €	3 años	8,63 €
TeamGantt	0 €	3 años	0 €
Mendeley Desktop	0 €	3 años	0 €
Trello	0 €	3 años	0 €
Total	**235,99 €**		**29,51 €**

Tabla 3: Presupuesto de software

En este apartado se podría utilizar otras herramientas como Google Drive para ahorrar el coste de Microsoft Office, pero como Office es la herramienta utilizada dentro de la empresa y se utiliza para la comunicación (Teams, Outlook), se ha preferido utilizarla.

6.1.3 Presupuesto de recursos humanos

De las 478 horas estimadas de dedicación al proyecto, una cantidad se realizarán con el rol de gerente del proyecto y las restantes, como desarrollador. Dentro de estas horas hay que calcular el tiempo de las reuniones que se realizarán al final de cada tarea, que se tendrá en cuenta dentro de la tarea.

En la podemos ver una estimación de los costes de los recursos humanos. Los precios/hora mostrados en la están basados en el estudio de remuneración de Michael Page para el año 2020.

Rol	Precio por hora	Horas	Salario	Seguridad Social (34%)	Precio final
Director	23,67 €	17	402,39 €	136,81 €	539,20 €
Jefe de proyecto	18,93 €	112	2.120,16 €	720,85 €	2.841,01 €
Desarrollador	11,36 €	349	3.964,64 €	1.347,98 €	5.312,62€
Total		**478**	**6.487,19€**		**8.692,83 €**

Tabla 4: Presupuesto de recursos humanos

Una vez realizado el cómputo de horas de cada rol y su precio, es coherente hacer un cálculo por tareas, ya que así se podrá ver la participación de cada rol dentro de cada una de las tareas, así como el precio de cada una de éstas. Dicho cálculo lo podemos ver en la *Tabla 5*.

Tarea	Duración (h)	Dedicación (h)			Coste
		Director	Jefe de proyecto	Desarrollador	
PPO	80	4	67	9	1.465,23 €
PPO-A	40	2	35	3	743,97 €
PPO-G	40	2	32	6	721,26 €
ADW	54	3	5	46	688,22 €
ADW-D	10	1	1	8	133,48 €
ADW-K	22	1	2	19	277,37 €
ADW-I	22	1	2	19	277,37 €
LAT	30	1	5	24	390,96 €
ABD	54	3	5	46	688,22 €
ABD-D	10	1	1	8	133,48 €
ABD-L	22	1	2	19	277,37 €
ABD-K	22	1	2	19	277,37 €
OBD	141	3	10	128	1.714,39 €
OBD-C	54	1	4	49	656,03 €
OBD-T	54	1	4	49	656,03 €
OBD-TC	33	1	2	30	402,33 €
SBD	98	2	10	86	1.213,60 €
SBD-T	44	1	4	39	542,43 €
SBD-TS	54	1	6	47	671,17 €
RPO	21	1	10	10	326,57 €
Total	**478**	**17**	**112**	**349**	**6.487,19 €**

Tabla 5: Estimación de presupuesto y tiempo por tarea

6.1.4 Costes indirectos

La *Tabla 6* muestra una estimación de los costes indirectos que no están categorizados en ninguna de las secciones anteriores. En esta tabla se ha considerado el caso peor, es decir, la luz encendida durante todo el día y el ordenador conectado durante todo el desarrollo del proyecto.

Producto	Precio	Unidades	Coste estimado
Consumo ordenador	0.133 €/ kWh	0,238 kWh * 478h	15,13 €
Fibra	34,91 €	4,5 meses	157,10 €
Luz	0.133 € / kWh	0,240 kWh * 478h	15,26 €
Alquiler	95 € / mes	4,5 meses	427,5 €
Total			**614,99 €**

Tabla 6: Costes indirectos

6.1.5 Costes imprevistos

Como se ha comentado en anteriores secciones, el tiempo de duración de las tareas puede sufrir desviaciones y, por lo tanto, tendrá repercusiones sobre el presupuesto. Las desviaciones que pueden suceder son las comentadas ya en la sección 5.6 que se pueden resumir en dos puntos: que la realización de la tarea ha llevado menos tiempo de lo esperado, que es poco probable y la otra opción es que la realización de la tarea lleve más tiempo del esperado, algo que es más probable.

En el primer caso, se disminuiría el presupuesto, ya que conllevaría a finalizar la tarea y continuar con las siguientes. En el segundo caso, se aumentaría el presupuesto, aunque se tiene un margen que son las 36 horas extras que se han estimado para posibles desviaciones. Para el cálculo del presupuesto se va a poner el caso en que la desviación fuese de una semana entera y fuesen horas de desarrollador, ya que al final es quien va a tener que recuperar esos retrasos.

También se puede encontrar la necesidad de disponer de nuevos recursos de software en el desarrollo del proyecto, aunque no es algo que sea muy probable que suceda. En cualquier caso, si sucediera, se intentaría optar por herramientas de licencia gratuita con funcionalidad similares a su opción de pago, por lo tanto, no supondría un incremento del presupuesto.

En recursos de hardware, el único inconveniente que podría surgir sería una avería del ordenador que utiliza para el desarrollo del proyecto, que podría producir tanto retrasos en las fechas de entrega como incremento del presupuesto para poder realizar las reparaciones necesarias. En este caso, para el cálculo se va a poner el peor caso posible, que hubiese que reemplazar el equipo.

Imprevisto	Precio	Probabilidad	Coste estimado
Tiempo	11,36 € * 29h	10%	32,94 €
Hardware	1.800 €	5%	90,00 €
Software	0 €	15%	0,00 €
Total			**122,94 €**

Tabla 7: Costes imprevistos

6.1.6 Presupuesto total

Sumando todos los presupuestos calculados en las secciones anteriores, se puede calcular el presupuesto total del proyecto, el cual se muestra en la *Tabla 8*. A este valor, se le añade un 5% de contingencias para cubrir gastos inesperados que puedan surgir durante el transcurso del proyecto. Se ha añadido únicamente un 5% de contingencia ya que se considera que todos los imprevistos que pueden surgir se han tenido en cuenta, y, por lo tanto, es poco probable que surja un imprevisto no tenido en cuenta.

Concepto	Coste estimado
Recursos Hardware	1.800,00 €
Recursos Software	235,99 €
Recursos Humanos	8.692,83 €
Recursos indirectos	614,99 €
Costes imprevistos	122,94 €
Subtotal	**11.466,75 €**
Contingencia (5%)	573,34 €
Total	**12.040,09 €**

Tabla 8: Presupuesto total

Una vez realizado este presupuesto, habrá que marcar ciertas pautas para poder tener un control de este presupuesto y poder detectar así desviaciones o contemplar alternativas.

6.2 Control del presupuesto

Para poder realizar una valoración económica de cada una de las tareas, se utilizará Microsoft Teams con la extensión de Trello para poder ir apuntando las horas empleadas en la tarea, así se podrán calcular las horas reales empleadas en la tarea y se podrá mirar si ha habido alguna desviación comparándolas con las horas estimadas. Con este cálculo de la desviación de horas empleadas en la tarea, se podrá calcular la desviación económica que se ha producido aplicando el precio por hora de cada uno de los roles.

Además, para posibles desviaciones, se ha realizado el cálculo del presupuesto de imprevistos, con la probabilidad de que suceda cada uno de ellos y también se ha calculado un presupuesto de contingencia para aquellos imprevistos que no se han tenido en cuenta a la hora de realizar la planificación.

6.3 Reflexión sobre sostenibilidad

6.3.1 Dimensión ambiental

Para cuantificar el impacto ambiental de la realización del proyecto se van a utilizar los KwH como unidad de medida. Con ello, representaremos el consumo energético de los recursos que intervienen en la ejecución del proyecto.

Se utilizarán las estimaciones de tiempo de la *sección 5.3* y los recursos detectados en la sección 5.5. Con todos estos datos, podemos ver el consumo en la *Tabla 9*.

Recurso	Consumo	Horas	Consumo total
MacBook Pro	238 W	478	109,48 kWH
Director	1,6 W	17	0,03 kWH
Jefe de proyecto	1,6 W	112	0,18 kWH
Desarrollador	1,6 W	349	0,56 kWH
Luz	240 W	478	110,40 kWH
Total			**220,62 kWH**

Tabla 9: Estimación del consumo energético

Para el cálculo se ha considerado el caso peor, es decir, teniendo el portátil siempre conectado a la corriente y teniendo la luz encendida durante todo el proyecto.

Para intentar disminuir este coste, se intentará trabajar en horas que se pueda trabajar con luz natural y conectar el portátil únicamente cuando sea necesario.

En cuanto a las arquitecturas resultado del proyecto, el consumo energético variará según el número de máquinas que se requieran y cuánta electricidad consumen.

Una forma de optimizar el consumo de estos sistemas será encontrando el número de nodos mínimos que producen más rendimiento conjuntamente, ya que en este tipo de arquitecturas llega un momento que por más máquinas que se añadan no va a aumentar el rendimiento sino al revés, como demuestra la ley de escalabilidad universal, y, además, se van a malgastar recursos tanto económicos como energéticos.

En las arquitecturas actuales del mercado, ya sean de DW o Big Data sucede lo que se acaba de comentar, que llega un punto en que el aumentar el número de máquinas no va a aumentar el rendimiento de la arquitectura, por lo tanto, se podría decir que la solución proporcionada en este proyecto no va a mejorar a las arquitecturas actuales desde el punto de vista ambiental, ya que el rendimiento de la arquitectura no va a ser la máxima prioridad del proyecto.

6.3.2 Dimensión económica

En la sección 6.1 se ha proporcionado una descripción detallada de los costes que tendrá el proyecto, teniendo en cuenta tanto recursos materiales como humanos.

Se podría reducir gastos reduciendo los recursos software con licencias de pago y utilizando alternativas de licencia gratuita, como en el caso de Microsoft Office ya comentado anteriormente. También se podría utilizar un portátil de menos coste ya que la realización del proyecto no exige de componentes potentes al portátil, pero se ha optado por utilizar el que ya tenía el desarrollador.

En cuanto a los sistemas resultantes del proyecto, se estima que los recursos hardware tienen 4 años de vida útil y los recursos software tiene 3 años de vida útil, por lo tanto, aunque la velocidad en la que evolucionan las tecnologías es más rápida que la vida útil de un recurso, se intentaría estirar al máximo la vida útil de éstos para llegar al máximo posible y así ahorrar costes.

Para calcular la viabilidad económica se considerarán como ventas los proyectos que se podrán conseguir gracias a la especialización del personal del departamento en los campos que trata el proyecto. Con el paso del tiempo se considerará que los proyectos han finalizado satisfactoriamente y, por lo tanto, el equipo habrá ido ganando experiencia y reputación, consiguiendo así más y mejores oportunidades.

Se ha elegido para la realización de este proyecto un cierre de costes anual y el estudio se realizará para un periodo de 4 años. Con todo esto, el cálculo del flujo de caja neto es el siguiente:

	Año 1	Año 2	Año 3	Año 4
Ventas	1.806,01	3.010,02	5.418,04	7.826,06
Gastos variables	0,00	0,00	0,00	0,00
Margen bruto	**1.806,01**	**3.010,02**	**5.418,04**	**7.826,06**
Costes indirectos	614,99	614,99	614,99	614,99
EBITDA	**1.191,02**	**2.395,03**	**4.803,05**	**7.211,07**
Amortizaciones	528,66	528,66	528.66	450,00
BAIT = EBIT	**662,36**	**1.866,37**	**4.274,39**	**6.761,07**
Impuestos	0,00	0,00	0,00	0,00
EBIAT	**662,36**	**1.866,37**	**4.274,39**	**6.761,07**
Amortizaciones	528,66	528,66	528,66	450,00
Flujo de Caja Operativo	**1.191,02**	**2.395,03**	**4.803,05**	**7.211,07**
Inversiones en activo fijo	0,00	0,00	0,00	0,00
Inversiones en capital de trabajo	0,00	0,00	0,00	0,00
Flujo de Caja Neto	**1.191,02**	**2.395,03**	**4.803,05**	**7.211,07**

Tabla 10: Cálculo del flujo de caja neto

Para la tasa de interés se ha escogido un 8,13% anual debido a que es lo que aplican las entidades de crédito en España para operaciones de 1 a 5 años. Con esta tasa de interés y los datos recogidos de la , el cálculo del VAN es el siguiente:

i	Inversión	CFF	t	CFF/(1+i)^t	Valor actual acumulado
8,13%	12.040,09		0	12.040,09	
		1.191,02	1	1.101,47	1.101,47
		2.395,03	2	2.048,42	3.149,89
		4.803,05	3	3.799,08	6.948,97
		7.211,07	4	5.274,91	12.223,88
				VAN	**183,79**

Tabla 11: Cálculo del VAN

Finalmente, y debido a que el VAN > 0 en un periodo de 4 años, se concluye que el proyecto es viable.

6.3.3 Dimensión social

La ejecución del proyecto va a permitir dotar de los conocimientos sobre los temas de estudio sobre todo al desarrollador, que va a ser quien tenga que investigar y desarrollar la redacción del documento.

En el proyecto se van a poder ver las arquitecturas de DW más importantes y las arquitecturas Big Data más populares, y tras ver estos dos mundos separadamente, se va a poder ver el estudio que une estos dos mundos, cosa no muy popular ya que la gente cree que son dos mundos totalmente opuestos.

A su vez, va a ayudar a las empresas a ver que los sistemas actuales que tienen si se utilizan para los requisitos que se han marcado en secciones anteriores, no acaban de funcionar correctamente y ver qué sistemas pueden utilizar para cumplir con esos requisitos.

Por último, se podrá ver un punto muy importante en los últimos tiempos, sobre todo por la salida del GDPR. La seguridad de los datos dentro de una empresa tiene que ser gestionada por el CIO[8] o el CTO[9] de cada empresa, que organizará y dará instrucciones y reglas a su equipo para que realicen la securización de los datos de la empresa.

[8] Chief Information Officer: Responsible de los sistemas de tecnologías de la información de una empresa.
[9] Chief Technical Officer: Responsible técnico del desarrollo y el correcto funcionamiento de los sistemas de información de una empresa.

7 Arquitecturas tradicionales de DW

En la década de los 90, a medida que las empresas se expandían a nivel mundial y la competencia se volvía más feroz, los ejecutivos de éstas se desesperaban por obtener información para mantenerse competitivos y mejorar los resultados. Los sistemas operacionales proporcionaban información para ejecutar las operaciones diarias pero lo que los ejecutivos necesitaban eran otros tipos de información que pudiesen utilizar a la hora de tomar decisiones estratégicas. Los responsables de la toma de decisiones querían saber en qué regiones geográficas enfocarse, qué líneas de productos potenciar y qué mercados fortalecer. Necesitaban el tipo de información con el contenido y el formato adecuados que pudiera ayudarles a tomar esas decisiones estratégicas. Los sistemas operacionales, por importantes que fueran, no podían proporcionarles la información estratégica, por lo que las empresas se vieron obligadas a recurrir a nuevas formas de obtener esa información.

Se necesitaba un nuevo tipo de entorno con el fin de proporcionar información estratégica para el análisis, las tendencias más exigentes y el monitoreo del rendimiento: el DW.

El DW es un sistema utilizado para poder generar informes y análisis de datos, y se considera un componente central de la inteligencia empresarial. Los DW son repositorios centrales de datos integrados de una o más fuentes de datos diferentes. Almacenan datos actuales e históricos en un solo lugar que se utilizan para crear informes analíticos para los trabajadores de toda la empresa.

Los datos almacenados en el DW se cargan desde los sistemas operacionales. Los datos pueden pasar a través de un almacén de datos operacional y pueden requerir la limpieza de los datos para garantizar la calidad de éstos antes de que se usen para la realización de informes.

Bill Inmon, considerado el padre del DW, proporcionó la siguiente definición del concepto: “Un DW es una colección de datos orientada al tema, integrada, no volátil y variante en el tiempo para dar soporte a decisiones de dirección.”

De esta definición podemos extraer las cuatro características fundamentales de un DW:

- Orientado al tema: Un DW guarda los datos en un modelo que sigue conceptos de negocio a diferencia de los procesos de negocio. Los sistemas transaccionales siguen los procesos y, por lo tanto, solo deben incluir la mínima cantidad de datos sobre cualquier tema que sea necesario para soportar ese proceso. Un DW se ocupa de la definición completa de un determinado contexto empresarial o sector. El modelo siempre debe reflejar el concepto de negocio en la definición más completa posible.

  Por ejemplo, para una compañía de seguros, la parte transaccional contendrá información sobre automóviles, vida, salud y accidentes y en cambio, las principales áreas temáticas del DW pueden ser los clientes, las pólizas, las primas y las reclamaciones.

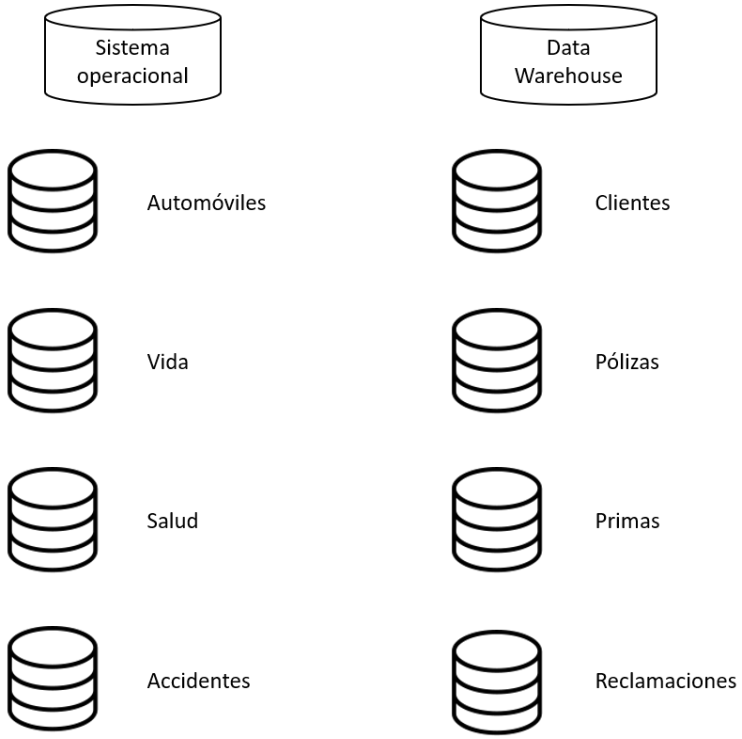


Ilustración 3: Ejemplo característica de un DW *- Orientado al tema*

- Integrado: Los datos provienen de una cantidad de fuentes de datos diversa y el modelo lógico del DW tiene que ser integrado y consistente. El formato de los datos y la utilización de los valores está estandarizado. Además, se deben minimizar las representaciones de datos repetidos o redundantes. El modelo lógico debe e incluso tiene que abordar la eliminación de datos redundantes.

  Un ejemplo sería que en una empresa se está guardando información sobre si los clientes son hombres o mujeres y esta información viene de diferentes aplicaciones en formas diferentes: en la aplicación 1 se guarda hombre, mujer como h, f, en la aplicación 2 como 1, 0 y en la aplicación 3 como hombre, mujer. En la aplicación final se quiere guardar como hombre, mujer, por lo tanto, se transformarán los datos de la aplicación 1 y 2 al pasarlos al DW para que sean hombre, mujer.

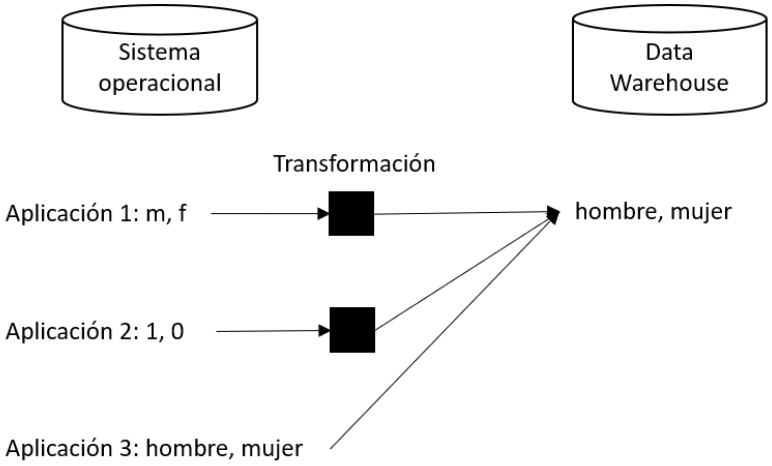


Ilustración 4: Ejemplo característica de un DW *– Integrado*

- No volátil: Un DW no cambia con las actualizaciones transaccionales. Esto es especialmente importante en el DW casi en tiempo real. Una vez que un registro se escribe en un DW, está completo. Es decir, no se elimina a menos que sea un error de carga real. Los errores que se guardan en el DW desde un sistema transaccional una vez escritos no deben eliminarse. Es responsabilidad del arquitecto de la *ETL*[10] asegurarse de que las reglas de escritura de datos en el DW siguen la definición de negocio correcta.

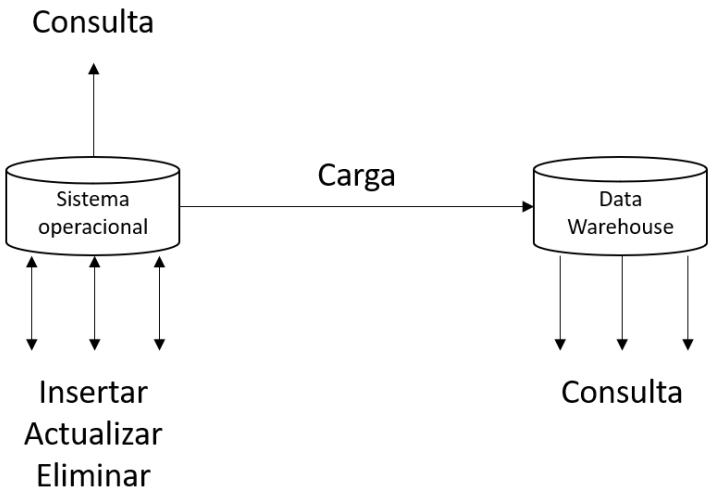


Ilustración 5: Ejemplo característica de un DW *- No volátil*

- Variable en el tiempo: Significa que el proceso de integración de los datos reconoce cuando dado un registro en un sistema transaccional exhibe cambios y crea una segunda versión del registro (o tercera, y así sucesivamente). Esto está ligado directamente con el problema de la volatilidad. El trabajo del DW es capturar cambios en los datos, pero también hacer un seguimiento de los datos a medida que cambian en el tiempo.

Ilustración 6: Ejemplo característica de un DW *- Variable en el tiempo*

[10] Extract, Transfrom and Load, en español, extraer, transformar y cargar.

Una vez contextualizado y definido el concepto de DW, vamos a explicar dos de las arquitecturas más importantes, la arquitectura presentada por Ralph Kimball y la arquitectura presentada por W.H. Inmon.

7.1 Arquitectura Kimball

Esta arquitectura también se conoce como Ascendente (*Bottom-up*), ya que al final el DW de la empresa no es más que la unión de los diferentes *Data Marts*, que están estructurados de una forma común a través de la estructura de bus. Esta característica hace que esta arquitectura sea más flexible y sencilla de implementar, ya que se puede construir un Data Mart como primer elemento del sistema de análisis y luego ir añadiendo otros que compartan las dimensiones que ya se han definido en el primero o incluir nuevas. En este sistema, el proceso ETL va guardando la información dentro de cada uno de los Data Mart de forma individual, respetando la estandarización de las dimensiones.

7.1.1 Modelo de datos

Antes de comenzar a hablar de la arquitectura, se considera conveniente explicar el concepto de modelo dimensional, ya que va a ser la base del sistema.

El modelado dimensional está ampliamente reconocido como la técnica más escogida para presentar datos analíticos porque cumple simultáneamente dos requisitos: proporciona datos que son entendibles por los usuarios del negocio y proporciona rendimiento rápido en las consultas.

Existen dos maneras de implementar un modelo dimensional. Si éste es implementado en sistemas de gestión de bases de datos relacionales se le conoce como esquema en estrella debido a su estructura similar a una estrella. En cambio, los modelos dimensionales en entornos de bases de datos multidimensionales son conocidos como *OLAP*[11] *cubes,* como podemos ver en la Ilustración 7.

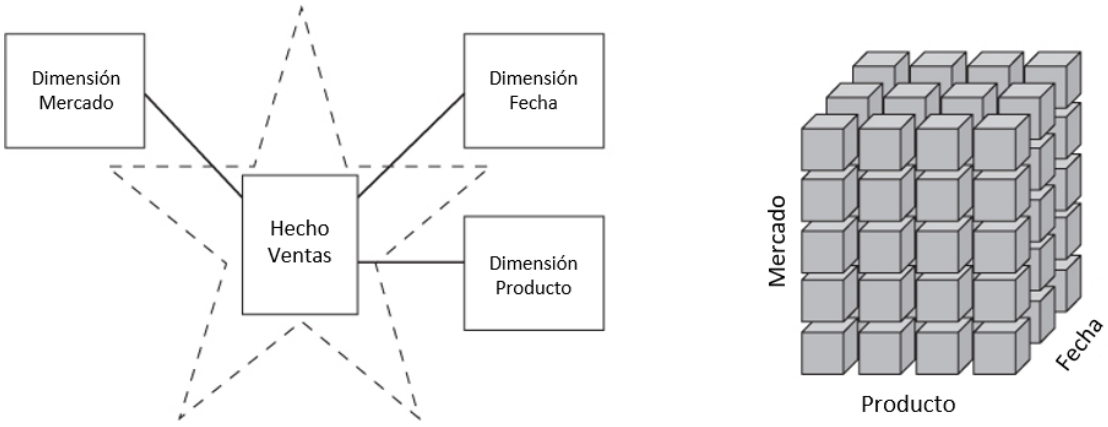


Ilustración 7: Esquema en estrella y esquema de cubos OLAP.
Inspirado en The data warehouse toolkit: the definitive guide to dimensional modeling

Una tabla de hechos en un modelo dimensional almacena medidas de rendimiento resultantes de eventos de los procesos de negocio de una organización. En cambio, una

[11] Online Analytical Processing, en español, procesamiento analítico en línea.

tabla de dimensiones contiene el contexto textual asociado a un evento de medida de proceso de negocio. Las tablas de dimensiones describen el “quién, qué, dónde, cuándo, cómo y por qué” asociado a un evento.

7.1.2 Componentes

Dada la pequeña explicación del modelado dimensional, ya se puede explicar la arquitectura. Dentro de la arquitectura Kimball hay cuatro componentes diferenciados: fuentes de datos operacionales, sistemas ETL, área de presentación de los datos y aplicaciones BI, los cuáles podemos ver en la *Ilustración 8*.

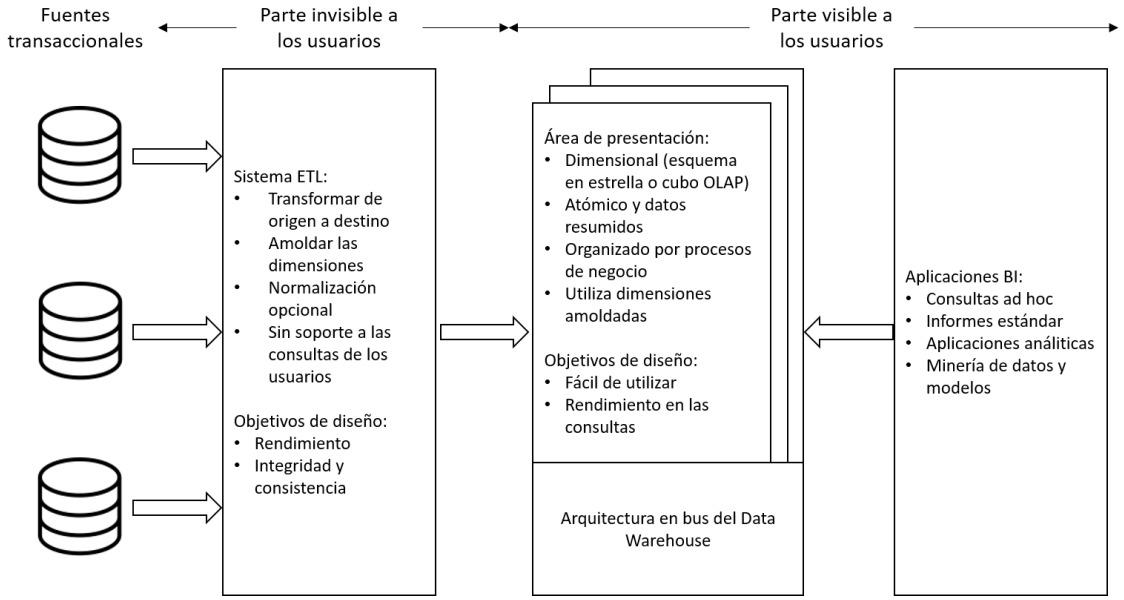


Ilustración 8: Elementos principales de la arquitectura Kimball
Inspirado en ‘The data warehouse toolkit: the definitive guide to dimensional modeling’

Viendo este esquema, se van a definir los diferentes componentes de la arquitectura.

- **Fuentes de datos operacionales**

Son los sistemas operacionales que capturan las transacciones del negocio. Hay que considerar como si estas fuentes estuviesen fuera del DW porque presuntamente no se va a tener o se va a tener muy poco control sobre el contenido y el formato de los datos que almacene. Las principales prioridades de estos sistemas son el rendimiento en el procesado y su disponibilidad. Las consultas que se realizan hacia estos sistemas son limitadas, las consultas de un único registro forman parte del flujo normal y son restringidas severamente por sus demandas a estos sistemas. Es seguro asumir que las fuentes de datos no serán consultadas de la manera amplia e inesperada en que se consulta los sistemas de DW.

Estos sistemas, guardan sólo un pequeño histórico de datos, ya que un buen DW puede liberar a estos sistemas de mucha responsabilidad de representar el pasado. En muchos casos, los sistemas origen son aplicaciones de propósito especial sin ningún compromiso de compartir datos comunes como productos, clientes, geografía o calendario con otros sistemas operacionales de la organización.

- **Sistema ETL**

El sistema ETL del entorno DW consta de un área de trabajo, estructuras de datos instanciadas y un conjunto de procesos. El sistema ETL es todo lo que hay entre las fuentes de datos operacionales y el área de presentación del DW.

La extracción es el primer paso en el proceso de obtener los datos dentro del entorno del DW. Extraer significa leer y comprender los datos de origen y copiar los datos necesarios en el sistema de ETL para su posterior manipulación. En este punto, los datos pertenecen al DW.

Una vez están los datos en el sistema ETL, existen numerosas transformaciones potenciales, como la limpieza de los datos (corregir errores ortográficos, tratar con nulos, …), combinar los datos de múltiples fuentes o eliminar duplicados. El sistema ETL añade valor a los datos con estas tareas de limpieza y amoldamiento al cambiarlos y mejorarlos. Además, estas tareas pueden ser diseñadas para crear metadatos de diagnóstico, lo que eventualmente lleva a la reingeniería de procesos de negocio para mejorar la calidad de los datos en las fuentes de datos a lo largo del tiempo.

El paso final del proceso es la estructuración física y la carga de datos dentro de los modelos dimensionales del área de presentación adecuados. Debido a que la misión principal del sistema ETL es entregar las tablas de dimensiones y hechos en el momento de la entrega, estos subsistemas son críticos. La mayoría de estos subsistemas se centran en el procesado de las tablas de dimensiones, como dividir o combinar columnas para presentar los valores adecuados. En contraste, las tablas de hechos suelen ser grandes y requieren mucho tiempo para cargarlas, pero prepararlas para el área de presentación suele ser sencillo. Cuando las tablas de dimensiones y hechos se actualizan, se indexan, se suministran con agregados apropiados y se garantiza la calidad, se notifica a la comunidad empresarial que se han publicado los nuevos datos.

- **Área de presentación**

El área de presentación del DW es donde los datos se organizan, almacenan y ponen a disposición de los usuarios, los redactores de informes y otras aplicaciones analíticas de BI para realizar consultas directas. Debido a que el sistema ETL está fuera de los límites, el área de presentación es el entorno DW para los usuarios, es todo lo que la empresa ve y toca a través de sus herramientas de acceso y sus aplicaciones de BI.

El área de presentación debe contener los datos estructurados con un modelo dimensional y estos datos deben ser atómicos y detallados. Los datos atómicos son necesarios para soportar asaltos de consultas imprevisibles de usuarios ad hoc. Aunque el área de presentación también puede contener datos agregados que mejoran el rendimiento, no es suficiente entregar estos agregados sin los datos granulares subyacentes en una forma dimensional. En otras palabras, es completamente inaceptable almacenar solo datos de resumen en modelos dimensionales mientras que los datos atómicos están bloqueados en modelos normalizados. No es práctico esperar que un usuario profundice en los datos dimensionales casi hasta el nivel más granular y luego pierda los beneficios de una presentación dimensional en el paso final. Los datos más detallados deben estar disponibles en el área de presentación para que los usuarios puedan hacer las preguntas más precisas posibles. Debido a que los requisitos de los usuarios son impredecibles y cambian constantemente, se debe proporcionar acceso a los máximos detalles para que puedan resumirse y responder las preguntas del momento.

El área de datos de presentación debe estar estructurado alrededor de eventos de medida de proceso de negocio. Este enfoque se alinea con los sistemas de captura de datos de las fuentes transaccionales. Los modelos dimensionales deben corresponder

a eventos de captura de datos físicos; no deben estar diseñados para entregar el informe del día. Los procesos de negocios de una empresa cruzan los límites de los departamentos y funciones de la organización. En otras palabras, se debe construir una tabla de hechos única para métricas de ventas atómicas en lugar de llenar bases de datos similares, pero ligeramente diferentes, que contengan métricas de ventas para los equipos de ventas, marketing, logística y finanzas.

Todas las estructuras dimensionales deben construirse utilizando dimensiones comunes y amoldadas. Esta es la base de la arquitectura de bus de almacenamiento de datos empresariales. La adhesión a la arquitectura de bus es la apuesta final para el área de presentación. Sin dimensiones compartidas y moldeadas, un modelo dimensional se convierte en una aplicación independiente. Los conjuntos de datos aislados que no se pueden unir son la ruina de los DW, ya que perpetúan puntos de vista de la empresa incompatibles.

El uso de las arquitecturas en bus es el secreto para construir sistemas de DW distribuidos. Cuando la arquitectura de bus se utiliza como base, se puede desarrollar el DW de la empresa de manera ágil, descentralizada, realista y de forma iterativa.

- **Aplicaciones BI**

El último gran componente de la arquitectura Kimball son las aplicaciones de BI. Por definición, todas las aplicaciones de BI consultan los datos en el área de presentación del DW. La consulta es el punto central del uso de datos para mejorar la toma de decisiones.

Una aplicación BI puede ser tan simple como una herramienta de consulta ad hoc o tan compleja como una sofisticada aplicación de modelado o minería de datos. Las herramientas de consulta ad hoc, tan poderosas como son, pueden ser entendidas y utilizadas de manera efectiva por solo un pequeño porcentaje de los potenciales usuarios de la compañía que utilizarán el DW. Es bastante probable que la mayoría de los usuarios accedan a los datos a través de aplicaciones y plantillas preconstruidas basadas en parámetros que no requieren que los usuarios construyan consultas directamente.

Con toda esta información, se termina la explicación de la arquitectura Kimball. Ahora veremos la otra gran arquitectura de DW: la arquitectura Inmon.

7.2 Arquitectura Inmon

Esta arquitectura también se conoce como *hub-and-spoke CIF*[12], y se puede ver cómo está organizada en la *Ilustración 9*.

[12] Corporate Information Factory, en español, Fábrica de Información Corporativa.

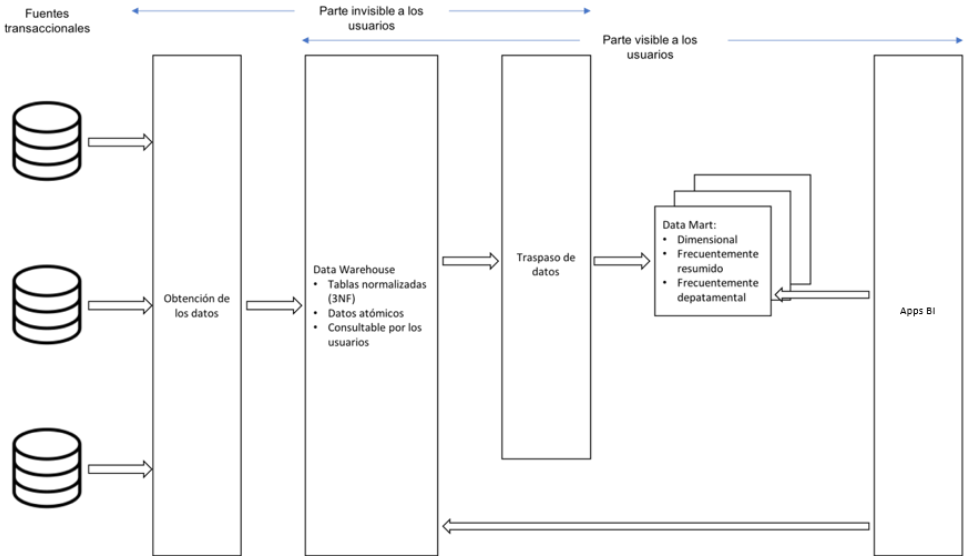


Ilustración 9: Arquitectura Inmon
Inspirado en 'The data warehouse toolkit: the definitive guide to dimensional modeling'

En esta arquitectura, los datos son extraídos de las fuentes transaccionales y procesadas a través del sistema ETL, al cual se refieren como Obtención de datos dentro de la *Ilustración 9*. Los datos atómicos que salen como resultado de este procesado son almacenados en una base de datos 3NF[13], la cual se conoce como Enterprise DW dentro de la arquitectura. Aunque la arquitectura Kimball permite opcionalmente la normalización para el procesamiento ETL, en la arquitectura Inmon es obligatorio tener el DW normalizado. Como la arquitectura Kimball, la arquitectura Inmon aboga por la coordinación y la integración de los datos empresariales. La diferencia está en que la arquitectura Inmon dice que el DW normalizado ya cumple con este rol, mientras que la arquitectura Kimball destaca la importancia de un bus empresarial con dimensiones conformes.

Las organizaciones que han adoptado la arquitectura Inmon a menudo tienen usuarios que acceden al repositorio del DW debido a su nivel de detalle o a la disponibilidad de los datos. Sin embargo, los procesos siguientes de ETL también pueblan la generación de informes y los entornos analíticos para apoyar a los usuarios de negocios. Aunque a menudo están estructuradas dimensionalmente, las bases de datos analíticas resultantes suelen diferir de las estructuras en el área de presentación de la arquitectura de Kimball, que a menudo están centradas en el departamento (en lugar de organizarse en torno a los procesos de negocios) y se rellenan con datos agregados (en lugar de detalles atómicos). Si los procesos ETL aplican reglas comerciales más allá del resumen básico, como el cambio de nombre de columnas o cálculos alternativos por parte del departamento, puede ser difícil vincular estas bases de datos analíticas con el repositorio atómico del DW.

Con la explicación dada de las dos arquitecturas se cree que es suficiente para poder hablar de las limitaciones que tienen.

[13] Tercera Forma Normal de una base de datos. Para serlo debe cumplir tres condiciones: estar en segunda forma normal, ningún atributo no-primario de la tabla es dependiente transitivamente de una clave primaria y es una relación que no incluye ningún atributo clave

8 Limitaciones de las arquitecturas de DW

La aparición de nuevas tecnologías como el Big Data, el NoSQL, *Data Science*, el *self-service* y las demandas de velocidad y agilidad hacen que el DW no sea la mejor opción. En esta sección hablaremos de los DW cuando los sometemos a grandes cantidades de datos, con gran variedad de fuentes de datos y latencias muy diversas.

- **Escalabilidad**

La infraestructura del DW aborda el volumen de datos como un problema de ampliación, es decir, pone como solución comprar más hardware a medida que los datos y las cargas de trabajo de procesamiento crecen. La ampliación es inadecuada para los volúmenes de datos, las cargas de trabajo de procesado y las tasas de consultas actuales. La infraestructura suele estar diseñada para funcionar a un nivel que satisfaga las demandas de las cargas de trabajo máximas. Como el volumen de datos, la complejidad de procesamiento y los casos de uso de los datos continúan expandiéndose, los picos de carga y los valles se hacen más extremos. Las herramientas para la carga de trabajo máximo son costosas y gran parte de la capacidad de computación no se utiliza.

- **Variedad de datos**

La mayoría de los DW están implementados utilizando sistemas de gestión de bases de datos relaciones. La tecnología relacional era la tecnología de bases de datos predominante en su día y la mayoría de los datos del DW eran procedentes de las bases de datos relacionales utilizadas por los sistemas transaccionales de las empresas. El fenómeno del Big Data amplió radicalmente la variedad de fuentes de datos disponibles y las formas en que los datos se organizan y estructuran. Las bases de datos modernas deben poder ingerir datos de bases de datos de grafos, almacenes clave-valor, almacenes de documentos, ficheros XML, ficheros JSON y una gran variedad de otras fuentes.

- **Latencia de datos**

Un DW típico adquiere la mayoría de los datos a través de procesos en lotes de ETL con actualizaciones periódicas, siendo las más comunes las actualizaciones semanales y diarias. Hay métodos para reducir los tiempos de latencia de los procesos en lotes, pero no se puede conseguir la entrega en tiempo real. A medida que la velocidad del negocio se acelera, los datos controlan la automatización de procesos y la transformación digital intensifica la dependencia de los datos. Estos factores se combinan para expandir y amplificar la demanda de datos en tiempo real y de muy baja latencia. Las bases de datos modernas deben poder ingerir y procesar datos con la frecuencia adecuada para cada fuente de datos. Los modos de ingesta deben abarcar desde ETLs programadas periódicamente como el procesamiento de flujo en tiempo real.

De estos problemas podemos sacar un factor común, que son las bases de datos relacionales y su obligación de preservar las características ACID[14]. Por lo tanto, una solución a este problema es intentar utilizar tecnologías NoSQL o Big Data que relajan estas propiedades y ofrecen un buen rendimiento para grandes cantidades de datos, variedad de datos y diferentes latencias de datos.

[14] Características de los parámetros que permiten clasificar las transacciones de los sistemas de gestión de bases de datos. Es un acrónimo de Atomicidad, Consistencia, Aislamiento y Durabilidad.

9 Arquitecturas Big Data

El Big Data es el análisis masivo de datos. Una cuantía de datos, tan sumamente grande, que las aplicaciones software de procesamiento de datos que tradicionalmente se venían usando no son capaces de capturar, tratar y poner en valor en un tiempo razonable. Igualmente, el mismo término se refiere a las nuevas tecnologías que hacen posible el almacenamiento y procesamiento, además de al uso que se hace de la información obtenida a través de dichas tecnologías.

Las principales características del Big Data se pueden clasificar en cuatro magnitudes, más conocidas como las cuatro V del Big Data, relativas a volumen, variedad, velocidad y veracidad. A pesar de ello, en los últimos años ya no se habla de las cuatro V sino de las siete V del Big Data:

- **Volumen**

  Con volumen, se hace referencia a la cantidad de datos que son generados cada segundo, minuto y días en nuestro entorno. Es la característica más asociada al Big Data, ya que hace referencia a las cantidades masivas de datos que se almacenan con la finalidad de procesar dicha información.

- **Velocidad**

  La velocidad se refiere a los datos en movimiento por las constantes interconexiones que realizamos, es decir, a la rapidez en la que son creados, almacenados y procesados en tiempo real.

- **Variedad de los datos**

  La variedad se refiere a las formas, tipos y fuentes en las que se registran los datos. Estos datos pueden ser datos estructurados y fáciles de gestionar como son las bases de datos, o datos no estructurados, entre los que se incluyen documentos de texto, correos electrónicos, datos de sensores, audios, videos o imágenes hasta publicaciones en redes sociales.

- **Veracidad de los datos**

  Con veracidad se refiere al grado de fiabilidad de la información recibida. Es necesario invertir tiempo para conseguir datos de calidad, aplicando soluciones y métodos que puedan eliminar datos imprevisibles que puedan surgir como datos económicos o comportamientos de los consumidores que puedan influir en las decisiones de compra.

- **Viabilidad**

  La inteligencia empresarial es un componente fundamental para la viabilidad de un proyecto y el éxito empresarial. Se trata de la capacidad que tienen las compañías en generar un uso eficaz del gran volumen de datos que poseen. Es necesario filtrar a través de estos datos y seleccionar cuidadosamente los atributos y factores que son capaces de predecir los resultados que más interesan a las empresas.

- **Visualización de los datos**

  Cuando se habla de visualización se hace referencia al modo en el que los datos son presentados. Una vez que los datos son procesados, necesitamos representarlos visualmente de manera que sean legibles y accesibles, para encontrar patrones y claves ocultas en el tema a investigar.

- **Valor de los datos**

  El dato no es valor. Tampoco se tiene valor por el mero hecho de recopilar grandes cantidades de datos. El valor se obtiene de datos que se transforman en información, ésta a su vez en conocimiento y éste, en acción o decisión. El valor de los datos está en que sean accionables, es decir, que los responsables de las empresas puedan tomar la mejor decisión en base a esos datos.

Como podemos observar analizando las diferentes características que poseen los sistemas Big Data es que estos sistemas tienen las características para cumplir los requisitos definidos en este proyecto: tratar con grandes cantidades de datos, con éstos proviniendo de una gran variedad de fuentes de datos y con latencias muy diversas (velocidad).

Por lo tanto, se va a indagar más en estos sistemas, ya que son una solución al problema planteado en el proyecto. Se va a realizar un estudio de las dos arquitecturas más importantes dentro del mundo Big Data: las arquitecturas Lambda y Kappa, de las cuales se hará un análisis de las características principales que poseen y las ventajas e inconvenientes de su utilización.

9.1 Arquitectura Lambda

La arquitectura Lambda es una arquitectura de procesamiento genérica. Pese a esto, posee ciertas características que la han hecho ser una de las arquitecturas más implementadas cuando se busca procesar grandes volúmenes de información. Algunas de estas características son que es escalable, es tolerante a fallos de dos tipos, humanos y de infraestructura, y está orientada a satisfacer las necesidades de un sistema robusto. Dicha arquitectura permite ejecutar una gran cantidad de cargas de trabajo y de casos de uso para los cuales son requeridas lecturas y escrituras de baja latencia.

La arquitectura Lambda está conformada por tres capas principales que son consideradas las responsables de realizar la ejecución de las tareas más relevantes del procesamiento de datos y las que entregan los resultados de dicho procesamiento: Capa de Lotes (*Batch Layer*), Capa de Servicios (*Serving Layer*) y Capa de Velocidad (*Speed Layer*).

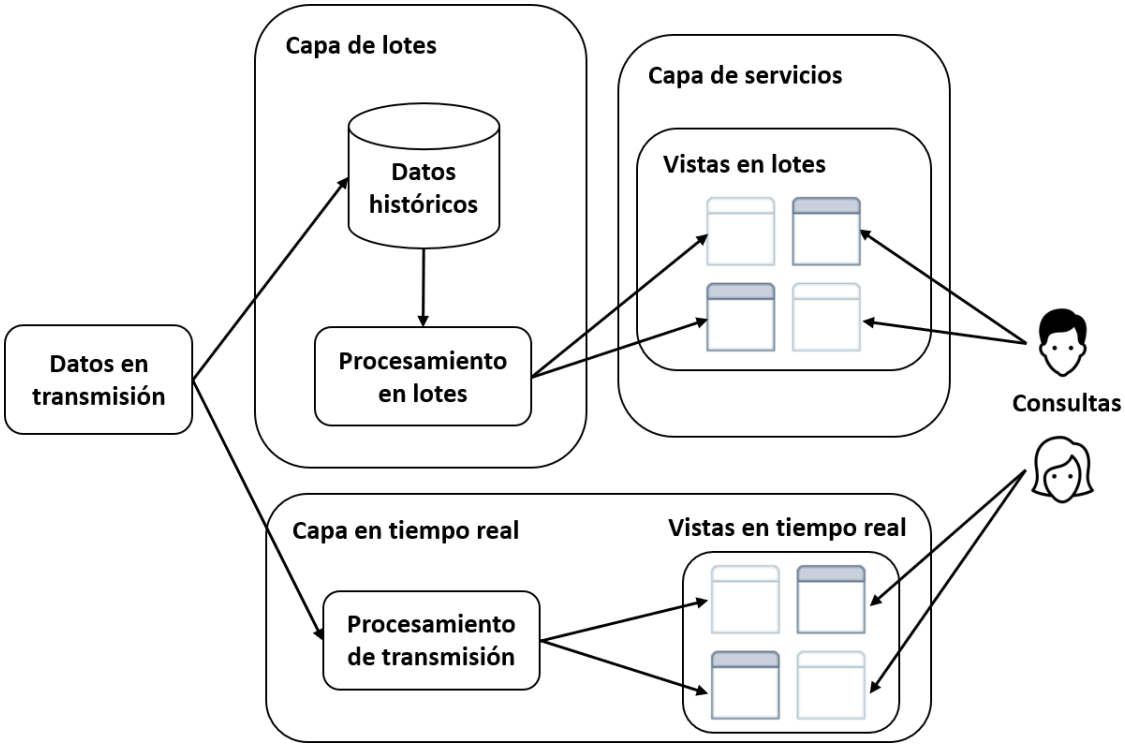


Ilustración 10: Arquitectura Lambda
Inspirado en https://www.paradigmadigital.com/techbiz/de-lambda-a-kappa-evolucion-de-las-arquitecturas-big-data/

Existen otros componentes que complementan la arquitectura desde el punto de vista de conectividad, aunque no son considerados como componentes principales. Sus módulos permiten la comunicación de los componentes de las capas con componentes externos o con los mismos conectores que hacen posible la ingesta de los datos que serán procesados.

La idea principal del funcionamiento de esta arquitectura es que toda la información que entra al sistema es replicada en la capa de velocidad y en la capa de lotes para que la información esté disponible para generar vistas en tiempo real en la capa de velocidad y vistas en lotes en la capa de servicios. De esta manera, cualquier consulta realizada al sistema puede ser resuelta combinando resultados de las vistas de ambas capas.

La **capa de lotes** recibe los nuevos datos, los combina con los datos históricos y vuelve a calcular los resultados iterando todo el conjunto de datos combinados. La capa de lotes opera sobre todos los datos y así permite al sistema conseguir los resultados más precisos posibles en forma de vistas. El inconveniente que tiene es que conseguir estos resultados tienen el coste de una gran latencia debido a su alto tiempo de computación. En resumen, esta capa tiene dos funciones principales: administrar el conjunto de datos histórico, que es inmutable y únicamente de almacenamiento y generar vistas en lotes que serán utilizadas por la capa de servicios.

La **capa de servicios**, por su parte, indexa las vistas en lotes generadas por la capa de lotes para que estén disponibles para ser consultadas con baja latencia y con consultas ad-hoc.

Por último, la **capa de velocidad** se utiliza para entregar resultados en baja latencia, cerca de tiempo real. La capa de velocidad recibe los datos que le llegan y realiza actualizaciones incrementales a los resultados de la capa de lotes para crear vistas en tiempo real. Gracias a los algoritmos incrementales implementados en esta capa, el tiempo de computación se reduce significativamente.

Las principales ventajas de utilizar una arquitectura Lambda en un sistema Big Data son:

- Los datos de entrada prevalecen intactos en una parte de almacenamiento inicial (*Raw Data*)
- Considera el problema de reprocesamiento de información cuando se producen cambios de criterio.

En cambio, las principales desventajas son:

- Alta complejidad operacional
- Limitaciones a la hora de la implementación, tiene que estar soportada por la intersección de los dos sistemas.
- Dificultad de abstraer paradigmas de programación diferentes construidos sobre sistemas distribuidos que ya son estables.

9.2 Arquitectura Kappa

Una de las más importantes motivaciones que llevaron a inventar la arquitectura Kappa fue evitar mantener dos códigos diferentes de las capas de lotes y de velocidad. Por lo tanto, la idea principal de esta arquitectura es administrar la capa de lotes y la capa de velocidad usando una única capa de procesamiento en tiempo real. Partiendo de este objetivo, la arquitectura Kappa sólo cuenta con dos capas: capa de tiempo real y capa de servicios.

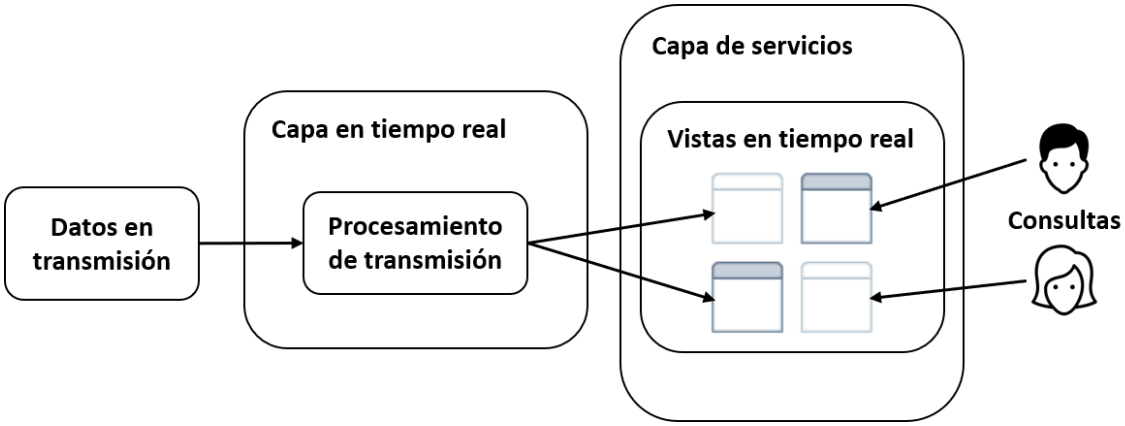


Ilustración 11: Arquitectura Kappa.
Inspirado en https://www.paradigmadigital.com/techbiz/de-lambda-a-kappa-evolucion-de-las-arquitecturas-big-data/

El reprocesado de los datos es un requisito importante para hacer visibles los efectos de los cambios de código en los resultados. La capa de tiempo real lanza tareas de procesamiento de los datos. Normalmente, una única tarea es lanzada para permitir el procesamiento de los datos en tiempo real. El reprocesado de datos únicamente se realiza cuando alguna parte del código de la tarea necesita ser modificada. Esto se consigue lanzando otra tarea modificando y replicando en ella todos los datos previos. Los resultados de esta nueva tarea se enviarán a una nueva tabla de salida y, cuando esta tarea haya finalizado el procesamiento de los datos, se deberán cambiar las aplicaciones para que consuman de la nueva tabla de salida. Finalmente, similarmente a la arquitectura Lambda, la capa de servicios es utilizada para consultar los resultados.

Las principales ventajas de utilizar la arquitectura Kappa en un sistema Big Data son:

- Solo se reprocesa cuando hay un cambio en el código.
- Solo hay que mantener un código.

Por otro lado, las desventajas son:

- Limitación del sistema a la hora de tratar con problemas complejos ya que no es realista creer que el código del procesado en lotes y el procesado en tiempo real sea el mismo, ya que el tiempo que se dispone en cada uno es distinto.

10 Orquestación de las arquitecturas Big Data

Tras haber realizado el estudio de las arquitecturas Big Data, se va a elegir una de ellas para poder continuar con el desarrollo del proyecto. Viendo las características y los componentes de ambas arquitecturas, se puede ver que la arquitectura Kappa flaquea a la hora de realizar aplicaciones complejas, ya que sólo se pueden realizar cálculos que se hagan durante el tiempo X que duren los intervalos del *streaming* que es la única forma en que nos llegan los datos en esta arquitectura. En la arquitectura Lambda, tenemos la Batch Layer que nos permite realizar cálculos complejos ya que no dependemos del tiempo en que nos llegan los datos. Por lo tanto, y viendo que lo que queremos es simular el comportamiento de un DW, en el que los cálculos pueden llegar a ser muy complejos, se elige la arquitectura Lambda para continuar con el proyecto.

Como ya se ha visto, la arquitectura Lambda está formada por la Batch Layer o capa de lotes, la Speed Layer o capa de velocidad y la Serving Layer o capa de servicios. De estas tres capas, para simular el comportamiento de un DW, este proyecto se va a centrar únicamente en la capa de lotes y en la capa de servicios, ya que es lo más similar a la arquitectura tradicional que se quiere simular. Con ello, la arquitectura que se va a utilizar para el análisis es la que podemos ver en la Ilustración 12.

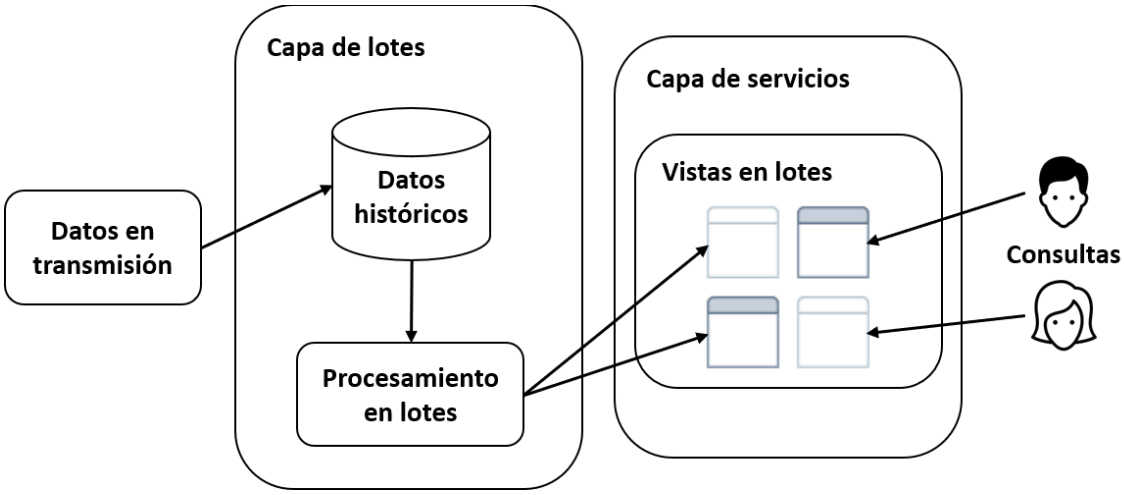


Ilustración 12: Arquitectura Lambda preparada para la orquestación

Viendo esta arquitectura y fijándonos en la arquitectura de Ralph Kimball (*Ilustración 8*), podemos hacer el mapeo diciendo que el componente de los datos en transmisión o streaming data haría de ETL, el de datos históricos o raw data sería el DW en sí, el procesamiento en lotes o *batch process* correspondería al acceso y el procesamiento de los datos para hacer visibles los datos a los usuarios y formaría parte del DW y la capa de servicios sería el almacén de datos con el cubo OLAP, como podemos ver en la Ilustración 13.

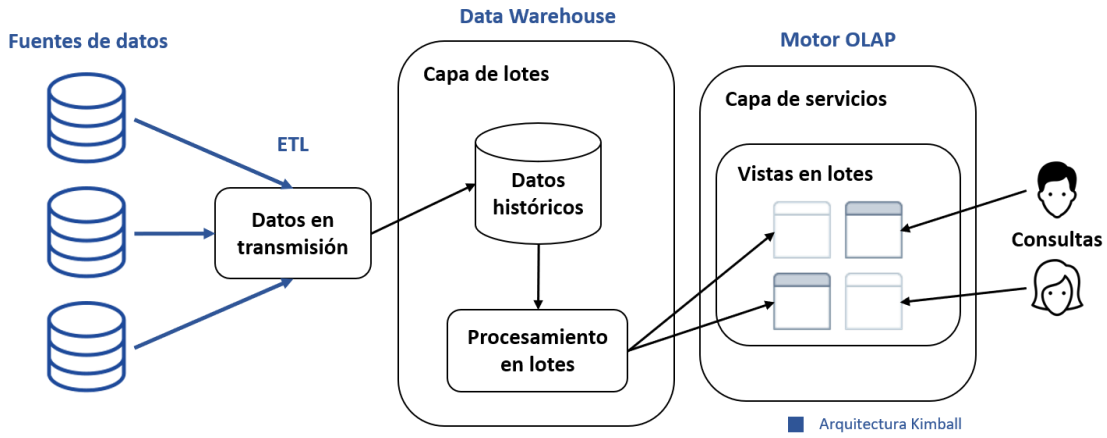


Ilustración 13: Paso de arquitectura Kimball a arquitectura Lambda

Por lo tanto, debemos ver que tecnologías se utilizarán dentro del almacén de datos históricos, el procesamiento en lotes y la capa de servicios, además de tecnología para controlar los datos en transmisión.

10.1 Tecnologías para la ingesta de datos en transmisión

En el paso de los datos en transmisión o streaming al almacén de datos históricos necesitamos una tecnología que nivele el paso de estos datos para que no se estén intentando guardar más datos de los que realmente puede soportar la herramienta que se utilice para almacenar esos datos. Existen diferentes tecnologías que puedan cumplir con estos requisitos, pero hay una que está actualmente por encima del resto y que es utilizada por la mayoría de las empresas: Apache Kafka.

10.1.1 Apache Kafka

Kafka es una plataforma software de procesamiento de flujo de código libre. El objetivo del proyecto es proporcionar una plataforma unificada, de alto rendimiento y baja latencia para el manejo de las fuentes de datos en tiempo real. Su capa de almacenamiento es esencialmente una cola de mensajes pub/sub muy escalable diseñada como un log de transacciones distribuida, lo que la hace altamente valiosa para las infraestructuras empresariales para procesar datos en streaming. Además, Kafka se conecta a sistemas externos (para importar/exportar datos) a través de Kafka Connect.

Kafka se basa en el *commit log*[15], y permite a los usuarios suscribirse y publicar datos en un número de sistemas o aplicaciones en tiempo real cualesquiera.

[15] Registro de confirmaciones, es decir, transacciones que se han realizado correctamente.

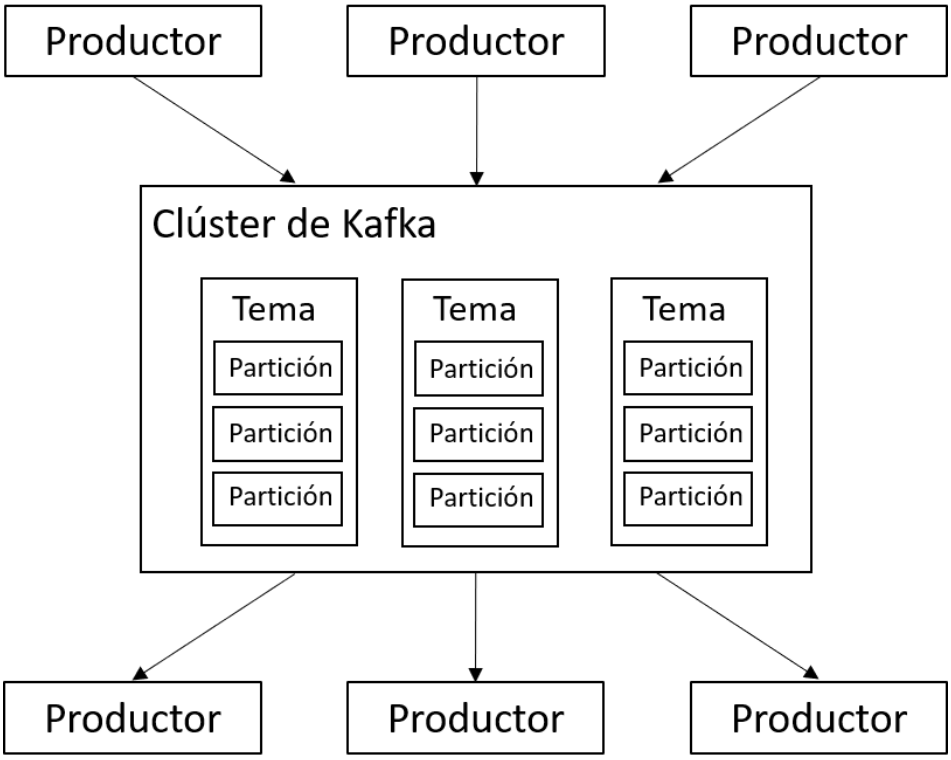


Ilustración 14: Arquitectura Apache Kafka
Inspirado en https*://en.wikipedia.org/wiki/Apache_Kafka*

Kafka almacena mensajes clave-valor que provienen de muchos procesos llamados productores o *producers.* Los datos se pueden dividir en diferentes particiones o *partitions* dentro de diferentes temas o *topics*. Dentro de una partición, los mensajes están estrictamente ordenados por sus *offsets*[16], y se indexan y almacenan junto con una marca de tiempo. Otros procesos llamados consumidores o *consumers* pueden leer mensajes de las particiones. Para el procesamiento de flujos, Kafka ofrece la API Streams que permite escribir aplicaciones Java que consumen datos de Kafka y escribir los resultados de nuevo en Kafka.

Kafka se ejecuta en un clúster de uno o más servidores (llamados intermediarios o *brokers*), y las particiones de todos los temas se distribuyen entre los nodos del clúster. Además, las particiones se replican a múltiples intermediarios. Esta arquitectura le permite a Kafka entregar flujos masivos de mensajes de forma tolerante a fallos y le ha permitido reemplazar algunos de los sistemas de mensajería convencionales como Java JMS[17], AMQP[18], etc.

Kafka soporta dos tipos de temas: regulares y compactos. Los temas regulares se pueden configurar con un tiempo de retención o un espacio enlazado. Si hay registros que son más antiguos que el tiempo de retención especificado o si se excede el límite de espacio para una partición, Kafka puede eliminar los datos antiguos para liberar espacio de almacenamiento. De forma predeterminada, los temas se configuran con un tiempo de retención de 7 días, pero también es posible almacenar datos de forma indefinida. En cuanto a los temas compactados, los registros no caducan en función de los límites de tiempo o espacio. En cambio, Kafka trata los mensajes posteriores como

[16] Posición de un mensaje dentro de una partición.
[17] Java Message Service.
[18] Advanced Message Queuing Protocol.

actualizaciones de mensajes más antiguos con la misma clave y garantiza que nunca se eliminará el último mensaje por clave. Los usuarios pueden eliminar los mensajes por completo escribiendo un valor nulo para una clave específica.

10.2 Tecnologías para el almacén de datos histórico

Para esta capa hay que tener en cuenta que los datos que provienen del streaming pueden venir en una gran variedad de formas de estructuración, por lo tanto, la tecnología que se utilice aquí debe ser capaz de soportar diferentes tipos de datos. Con este requisito, podemos descartar las bases de datos relacionales. También debe ser capaz de guardar los datos en el estado que vengan, es decir, sin realizar transformaciones en ellos, ya que pueden venir a grandes velocidades y si se realiza procesado, se podrían perder parte de los datos.

Las tecnologías que pueden soportar estos requisitos son los sistemas de archivos distribuidos, entre los cuales destaca el HDFS[19] y las bases de datos clave-valor, entre las cuales destacan HBase y Cassandra.

10.2.1 HDFS

El HDFS es un sistema de archivos distribuidos. Tiene muchas similitudes con los sistemas existentes, sin embargo, las diferencias con éstos son significativas. HDFS es altamente tolerante a fallos y está diseñado para implementarse en hardware de bajo coste. HDFS proporciona acceso de alto rendimiento a los datos de la aplicación y es adecuado para aplicaciones que tienen grandes conjuntos de datos. HDFS relaja algunos requisitos POSIX para permitir el acceso en streaming a los datos del sistema de archivos. HDFS se creó originalmente como infraestructura para el proyecto del motor de búsqueda web Apache Nutch y ahora es un subproyecto Apache Hadoop.

Sus principales características son:

- HDFS, es un sistema de ficheros que está especialmente diseñado para funcionar bien cuando se almacenan archivos grandes, que posteriormente se leerán de forma secuencial.

- HDFS no se comporta especialmente bien cuando lo que se pretende es realizar accesos aleatorios a los archivos, ni cuando estos se actualizan frecuentemente.

- Como se ha comentado, el hecho de que el sistema de ficheros sea distribuido proporciona ventajas, pues se pueden almacenar muchos más datos que los que se podrían almacenar en una sola máquina. Además, en general la cantidad de datos que se puede almacenar puede escalar con facilidad, ya que basta con añadir nuevos nodos al clúster para aumentar su capacidad de almacenamiento.

- HDFS proporciona redundancia, es decir, almacena los ficheros varias veces y en varios equipos distintos, para evitar que, si uno de ellos falla, los datos se pierdan. Esto, además, permite que se pueda emplear hardware relativamente económico para desplegar Hadoop, puesto que el sistema es tolerante a fallos.

[19] Sistema de archivos distribuidos de Hadoop.

Existen otros sistemas de ficheros distribuidos como GFS[20], en el cual se inspiró HDFS, GPFS[21] de IBM o GFS[22] de Red Hat, pero el más utilizado para la arquitectura Lambda y en general es HDFS, por lo tanto, este proyecto se decanta por él frente a los otros.

10.2.2 Apache HBase

Apache HBase es una base de datos NoSQL clave-valor que se ejecuta sobre HDFS. Las operaciones HBase se ejecutan en tiempo real en su base de datos en lugar de en los trabajos MapReduce. HBase se particiona en tablas, y las tablas se dividen a su vez en familias de columnas. Las familias de columnas, que deben declararse en el esquema, agrupan un determinado conjunto de columnas (las columnas no requieren una definición de esquema). Por ejemplo, la familia de columnas "mensaje" puede incluir las columnas "a", "desde, "fecha", "asunto" y "cuerpo". Cada par de clave-valor en HBase se define como una celda, y cada clave consta de clave de fila, familia de columnas, columna y tiempo. Una fila en HBase es una agrupación de asignaciones claves-valor identificadas por la clave de fila. HBase disfruta de la infraestructura de Hadoop y se escala horizontalmente utilizando servidores listos para usar.

HBase tiene una arquitectura centralizada en la que el servidor maestro es responsable de monitorear todas las instancias de RegionServer[23] en el clúster, y es la interfaz para todos los cambios de metadatos. Proporciona CP (Consistencia y Disponibilidad) del teorema CAP[24].

HBase está optimizado para lecturas, es compatible con la escritura única del nodo maestro y el modelo de consistencia estricta resultante, así como el uso de particionamiento ordenado que admite escaneos de filas, por lo que HBase es muy adecuado para hacer escaneos basados en rangos. Debido al particionamiento ordenado, HBase se escalará horizontalmente fácilmente y seguirá siendo compatible con los escaneos de filas por rangos.

HBase admite cuatro operaciones principales: 'poner' para insertar o actualizar filas, 'escanear' para recuperar un rango de celdas, 'obtener' para devolver celdas de una fila específica y 'eliminar' para eliminar filas, columnas o versiones de columnas de la tabla. El control de versiones está disponible para poder recuperar los valores anteriores de los datos (el historial se puede eliminar de vez en cuando para despejar el espacio a través de las compactaciones de HBase). Aunque HBase incluye tablas, solo se requiere un esquema para tablas y familias de columnas, pero no para columnas, e incluye funcionalidad de incremento/contador.

HBase no admite índices secundarios[25] de forma nativa, pero un caso de uso de disparadores es que un disparador en una operación 'poner' puede automáticamente mantener actualizado un índice secundario y, por lo tanto, no pone la carga en la aplicación.

[20] Google File System.
[21] General Parallel File System.
[22] Global File System.
[23] Responsable de servir y administrar regiones.
[24] Teorema que enuncia que imposible para un sistema de computación distribuido garantizar simultáneamente: Consistencia, Disponibilidad y Tolerancia al particionado.
[25] Es un índice cuya clave de búsqueda especifica un orden distinto del orden secuencial del archivo.

En resumen, sus principales características son:

- Almacén Big Data distribuido y escalable
- Fuerte consistencia
- Construido sobre HDFS
- CP en CAP

10.2.3 Apache Cassandra

Apache Cassandra es un sistema de gestión de bases de datos distribuidas NoSQL y basada en un modelo de almacenamiento clave-valor que ofrece disponibilidad continua, alta escalabilidad y rendimiento, seguridad sólida y simplicidad operativa al tiempo que reduce el coste general de mantenerla.

Cassandra tiene arquitectura descentralizada, es decir, cualquier nodo puede realizar cualquier operación y además proporciona AP (disponibilidad y tolerancia de partición) del teorema de CAP.

Cassandra tiene un excelente rendimiento de lectura en una sola fila siempre y cuando la consistencia eventual que ofrece sea suficiente para el caso de uso. Las lecturas del quórum de Cassandra, que se requieren para una consistencia estricta, naturalmente serán más lentas que las lecturas de HBase. Cassandra no admite consultas de filas basadas en rangos, lo que puede ser limitante en ciertos casos de uso. Cassandra es muy adecuada para admitir consultas de una sola fila o para seleccionar varias filas según un índice de valor de columna.

Si los datos se almacenan en columnas en Cassandra para admitir consultas por rango, la limitación práctica de un tamaño de fila es de 10 MB. Las filas más grandes que eso causan problemas con la sobrecarga de compactación y el tiempo.

Cassandra admite índices secundarios en familias de columnas donde se conoce el nombre de la columna (no en columnas dinámicas).

Sus principales características son:

- Alta disponibilidad
- Escalabilidad incremental
- Eventualmente consistente
- Compensaciones entre consistencia y latencia
- Administración mínima
- Sin punto único de fallo: todos los nodos son iguales en Cassandra
- AP en el teorema de CAP.

10.2.4 Comparación

Habiendo visto estas tres tecnologías, primero se tendrá que analizar las diferencias entre un sistema de ficheros como Hadoop y una base de datos, yendo al ejemplo más básico que serían los RDBMS[26] tradicionales, mirando desde distintos puntos de vista como el volumen de datos, la arquitectura, …

Si nos fijamos en el volumen de datos, los RDBMS funcionan mejor cuando el volumen de datos es bajo (en *gigabytes*), ya que cuando el tamaño de los datos es enorme, es decir, *terabytes* o *petabytes*, los RDBMS no dan los resultados deseados. Por otro lado, Hadoop funciona mejor cuando el tamaño de los datos es grande. Puede procesar y almacenar fácilmente gran cantidad de datos de manera bastante efectiva en comparación con los RDBMS.

En cuanto a arquitectura, Hadoop está formado por tres componentes centrales, HDFS, Hadoop MapReduce, que se explicará en la siguiente sección y Hadoop YARN (utilizado para administrar los recursos informáticos en clústeres de nodos). En cambio, los RDBMS poseen propiedades ACID, las cuales son responsables de mantener y garantizar la integridad y la precisión de los datos cuando una transacción tiene lugar en una base de datos.

Si se mira la variedad de datos soportada por cada tipo de sistema, Hadoop tiene la capacidad de procesar y almacenar toda variedad de datos, ya sean estructurados, semiestructurados o no estructurados, aunque normalmente se utiliza para procesar grandes cantidades de datos no estructurados. Al contrario, los RDBMS tradicionales se utilizan únicamente para gestionar datos estructurados y semiestructurados. Así que en este aspecto se podría decir que Hadoop supera a los RDBMS.

En términos de latencia o tiempo de respuesta, Hadoop tiene un rendimiento más alto, ya que se puede acceder más rápidamente a lotes de conjuntos de datos grandes que con los RDBMS tradicionales, aunque no puede acceder a un registro particular del conjunto de datos rápidamente. Por lo tanto, se dice que Hadoop tiene baja latencia, pero el RDBMS es comparativamente más rápido en la recuperación de la información de los conjuntos de datos. Se tarda muy poco tiempo en realizar la misma función siempre que haya una pequeña cantidad de datos.

Mirando la escalabilidad que proporciona cada sistema, se puede ver que los RDBMS proporcionan escalabilidad vertical, que significa que se pueden agregar más recursos o hardware, como memoria, a una máquina en el clúster. En cambio, Hadoop proporciona una escalabilidad horizontal, lo que significa que se añaden más máquinas a los clústeres.

En conclusión, si se necesitan guardar ficheros y hacer accesos por fichero, es necesario implementar un sistema de ficheros. Por otro lado, si se necesitan hacer consultar avanzadas sobre los datos, no a nivel de fichero, es mejor una base de datos. Una vez vista las diferencias entre sistemas de ficheros y bases de datos, quedaría realizar la comparación entre HBase y Cassandra.

Tratar de determinar cuál de las dos bases de datos es mejor para un cliente depende realmente del proyecto en el que se necesite. Cada una tiene sus ventajas y, en ocasiones, la elección dependerá simplemente de las preferencias personales para llevar a cabo el desarrollo de software

[26] Sistema de Gestión de Bases de Datos Relacionales

HBase y Cassandra se utilizan en gran medida para los mismos fines. Ambos son almacenes de datos distribuidos NoSQL que enfatizan las lecturas/escrituras escalables. Ambos afirman una escalabilidad lineal, lo que significa que la capacidad de almacenamiento y el rendimiento están directamente relacionados con el número de nodos que operan en el clúster.

Ambos también enfatizan la replicación, lo que significa que los datos se replican en diferentes nodos para evitar la pérdida de datos. De manera similar, ambos son tolerantes a las particiones de red, lo que significa que seguirán funcionando incluso si ciertos nodos se caen o no se comunican a través de una red (tolerancia a fallos).

HBase se utiliza cuando se quiere enfatizar la consistencia en las lecturas a gran escala. Cassandra cuando se desea alta disponibilidad. Cassandra requiere una configuración mínima con poca sobrecarga de administración, lo que facilita el inicio.

Si se trabaja mucho con MapReduce y procesado en batch, es mejor utilizar HBase por su relación directa con HDFS. Cassandra es una buena base de datos para lecturas de una sola fila y está optimizada para escrituras. Cassandra, además, ofrece más flexibilidad en cuanto a las compensaciones del teorema de CAP, ya que se pueden configurar niveles de consistencia mientras que HBase carece de estas configuraciones explícitas.

En la siguiente tabla podemos ver las características principales de cada una de ellas para poder compararlas más fácilmente.

	Apache HBase	**Apache Cassandra**
Modelo de base de datos	Clave-Valor	Clave-Valor
Licencia	Libre	Libre
Lenguaje de implementación	Java	Java
Esquema de los datos	Libre	Libre
Tipo de arquitectura	Centralizada	Descentralizada
Teorema CAP	CP	AP
Métodos de partición	Sharding[28]	Sharding
Métodos de replicación	Factor de replicación elegible	Factor de replicación elegible
Compatible con MapReduce	Sí	Sí
Consistencia	Consistencia fuerte	Consistencia eventual
Concurrencia	Sí	Sí
Índices secundarios	No	Sí, pero restringido

Tabla 12: Comparativa HBase y Cassandra

[28] Segmentación de los datos de forma horizontal, es decir, partir la base de datos principal en varias bases de datos más pequeñas repartiendo la información.

Como conclusión a la comparación, se puede ver que se utilizará HBase cuando se quiera garantizar una consistencia más fuerte y Cassandra cuando sea más importante la disponibilidad del sistema. Con esto, se puede ver que son totalmente complementarias y que depende del proyecto en el que se esté trabajando para escoger una u otra según convenga.

10.3 Tecnologías para el procesado en lotes de los datos

En esta capa se tiene que utilizar una tecnología que use los datos de la capa anterior, sea capaz de agrupar aquellos datos que pertenezcan a un mismo proceso de negocio, aplique el procesado necesario a esos datos y los guarde en la capa de servicios.

Las tecnologías a tener en cuenta para esta capa son Hadoop MapReduce y Apache Spark.

10.3.1 MapReduce

MapReduce es un modelo de programación para dar soporte a la computación paralela sobre grandes colecciones de datos en clústeres de máquinas. Por regla general, MapReduce se utiliza en aquellos problemas de computación concurrente entre los que se encuentran involucrados grandes conjuntos de datos que deben ser procesados por una gran cantidad de nodos, a los que se refiere de manera conjunta como clústeres o *grids*. El procesamiento paralelo puede ocurrir con el empleo de datos almacenados tanto en sistemas de ficheros como en bases de datos.

Respecto a su funcionamiento, comienza dividiendo las grandes cantidades de datos en partes más pequeñas que la tarea *map* procesa en paralelo y produce pares clave-valor como salida. La salida de la tarea map se utiliza como entrada para la tarea *reduce* de manera que todos los pares clave-valor con la misma clave vayan a la misma función reduce. Tras esto, se agrupa el conjunto de claves-valor en un conjunto más pequeño de pares clave-valor que es el resultado final.

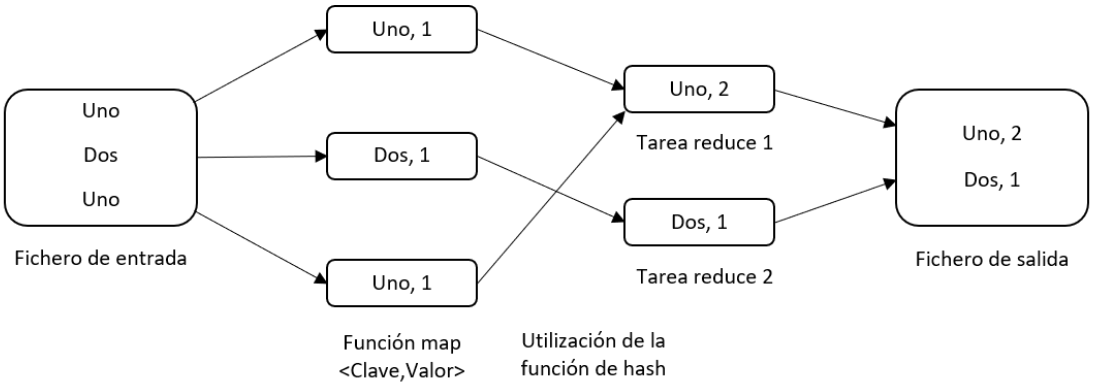


Ilustración 15: Ejemplo de funcionamiento de MapReduce.
Inspirado en https://dimensionless.in/what-is-the-difference-between-hadoop-and-spark/

10.3.2 Apache Spark

Spark es un *framework* de computación en clúster de código libre y proporciona una interfaz para la programación de clústeres completos con Paralelismo de datos implícito y tolerancia a fallos.

Spark tiene la base de su arquitectura en el llamado RDD[29], un conjunto de datos de solo lectura distribuidos a lo largo de un clúster de máquinas, que se mantiene en un entorno tolerante a fallos. Spark surgió como respuesta a las limitaciones del paradigma de computación en clúster de MapReduce, que obliga a una estructura de flujo de datos lineal particular en programas distribuidos.

Spark requiere de un administrador de clústeres y un sistema de almacenamiento distribuido. Para la gestión del clúster, se puede utilizar un clúster de Spark nativo, Hadoop YARN o Apache Mesos. Para el sistema de almacenamiento distribuido, Spark puede interactuar con una amplia variedad, que incluye HDFS, Cassandra, Kudu o se puede implementar una solución personalizada, entre otros.

10.3.3 Comparación

La diferencia clave entre MapReduce y Spark está en el enfoque de procesamiento, ya que Spark puede hacerlo en memoria, mientras que MapReduce tiene que leer y escribir en un disco. Como resultado, la velocidad de procesamiento difiere significativamente debido a que Spark puede ser hasta 100 veces más rápido. Sin embargo, el volumen de datos procesados también varía, MapReduce puede trabajar con conjuntos de datos mucho más grandes que Spark.

MapReduce no puede utilizarse para proporcionar resultados inmediatos, pero es muy adecuado para los datos recopilados durante un periodo de tiempo. Spark, en cambio, se puede utilizar tanto para el procesamiento batch como para el procesamiento de datos en tiempo real. Incluso si los datos se almacenan en un disco, Spark funciona más rápido.

Tanto MapReduce como Spark son de código libre, pero tienen costes de hardware asociados a ellos. Están diseñados para funcionar en hardware barato pero dado que MapReduce está basado en el funcionamiento en disco, requiere de discos más rápido mientras que Spark puede trabajar con discos estándar, pero requiere de una gran cantidad de *RAM*, por lo que es más caro.

El framework de programación de Spark es un mucho más sencillo que MapReduce. Sus *APIs* en Java, Scala, Python y R son fáciles de usar. MapReduce también tiene varios componentes que no requieren una programación compleja como Hive, Pig o Sqoop, que son fáciles de utilizar. Aunque en cuanto a lenguajes de programación, MapReduce solo soporta Java.

[29] Resilient Distributed Dataset

En la siguiente tabla podemos ver las características principales de cada una de ellas para poder compararlas más fácilmente.

	MapReduce	Apache Spark
Procesado de datos	Procesado en batch	Tanto procesado en batch como en tiempo real
Velocidad de procesado	Más lento que Spark debido a la latencia de disco	100 veces más rápido en memoria y 10 veces más rápido trabajando en disco
Coste	Menor coste en comparación a Spark	Más coste debido a la gran cantidad de RAM usada
Escalabilidad	Limitado a 1.000 nodos en un solo clúster	Limitado a 1.000 nodos en un solo clúster
Compatibilidad	Con la mayoría de fuentes de datos y formatos de fichero	Con fuentes de datos y formatos de fichero soportados por el clúster de Hadoop
Facilidad de uso	Bastante más complejo que Spark	Más fácil de utilizar debido a sus numerosas APIs
Soporte de lenguajes	Java	Java, Scala, Python y R
Complejidad	Difícil de escribir y depurar código	Fácil escritura y depuración de código
SQL	Soporte gracias a Hive Query Language	Soporte gracias a Spark SQL

Tabla 13: Comparativa MapReduce y Spark

10.4 Tecnologías para la capa de servicios

Dentro de esta capa necesitaremos el esquema en estrella y las operaciones OLAP básicas. Además, la tecnología que se utilice debe ser ágil en las consultas, por lo tanto, será un almacén de datos con particionamiento vertical, ya que mejora el rendimiento de las consultas.

En cuanto a los primeros requisitos (esquema en estrella y OLAP) tenemos tres opciones: encontrar una herramienta en que todo ello venga hecho, alguna herramienta que pueda dar alguna ayuda o realizarse encima de la tecnología que se escoja. Viendo estos tres puntos, sobre el primero no hay una tecnología que lo consiga, en el segundo podríamos hablar de Apache Druid y en el último si tenemos más opciones, pero se optará por analizar Apache Kudu.

10.4.1 Apache Druid

Druid es un almacén de datos distribuido, de código libre y con particionamiento vertical. Está diseñado para ingerir rápidamente cantidades masivas de datos y proporcionar consultas de baja latencia sobre los datos.

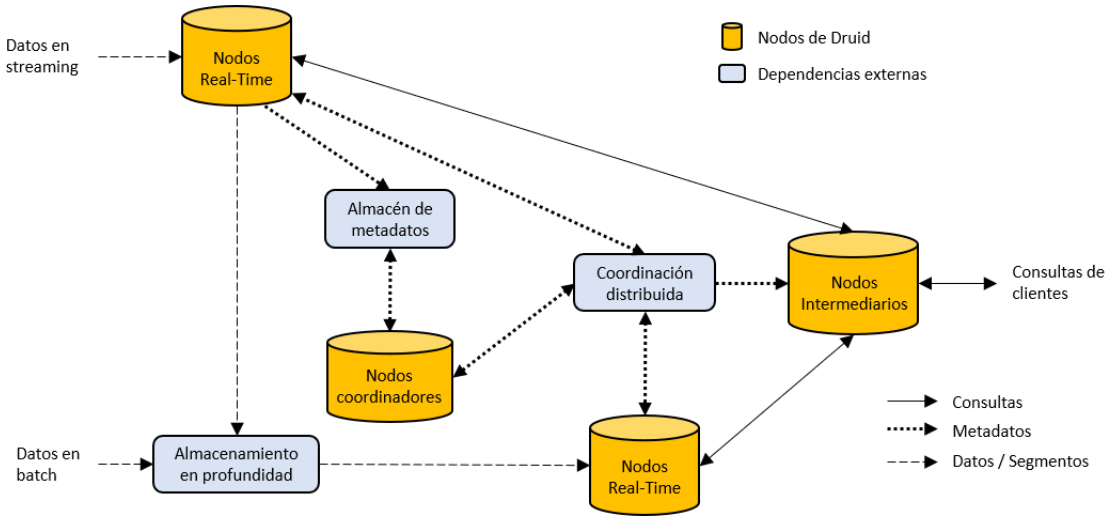


Ilustración 16: Arquitectura de Apache Druid.
Inspirado en https://dev.to/rusrushal13/learning-about-the-druid-architecture-184c

Una vez implementado, Druid se ejecuta como un clúster de procesos especializados (llamados nodos en Druid) para soportar una arquitectura tolerante a fallos donde los datos se almacenan de manera redundante y no hay un punto único de fallo del sistema. El clúster incluye dependencias externas para la coordinación (Apache ZooKeeper), almacenamiento de metadatos (por ejemplo, MySQL, PostgreSQL o Derby) y una instalación de almacenamiento profundo (por ejemplo, HDFS o Amazon S3) para la copia de seguridad de datos permanente.

Las consultas de los usuarios primero llegan a los nodos intermediarios, que los reenvían a los nodos de datos apropiados (históricos o en tiempo real). Dado que los segmentos de Druid pueden particionarse, una consulta entrante puede requerir datos de múltiples segmentos y particiones almacenados en diferentes nodos del clúster. Los nodos intermediarios pueden aprender que nodos tienen los datos requeridos y también combinar resultados parciales antes de devolver el resultado.

Las operaciones relacionadas con la gestión de datos en nodos históricos son supervisadas por nodos coordinadores. Apache ZooKeeper se utiliza para registrar los nodos y administrar ciertos aspectos de las comunicaciones entre nodos.

10.4.2 Apache Kudu

Kudu es un almacén de datos de código libre con fragmentación vertical del ecosistema de Apache Hadoop. Kudu comparte las propiedades técnicas comunes de las aplicaciones del ecosistema de Hadoop: se ejecuta en hardware básico, es escalable horizontalmente y admite operaciones de alta disponibilidad.

El diseño de Kudu es el que marca la diferencia y algunos de sus beneficios son:

- El procesamiento rápido de cargas de trabajo OLAP
- Integración con MapReduce, Spark y otros componentes del ecosistema de Hadoop
- Modelo de consistencia sólido pero flexible, que le permite elegir los requisitos de consistencia según la solicitud

- Rendimiento sólido para ejecutar cargas de trabajo aleatorias y secuenciales simultáneamente
- Alta disponibilidad. Los servidores y maestros de la tableta utilizan el algoritmo de consenso de balsa, que garantiza que mientras más de la mitad del número total de réplicas esté disponible, la tableta estará disponible para lectura y escritura. Por ejemplo, si están 2 de 3 réplicas o 3 de 5 réplicas, la tableta está disponible.

La *Ilustración 17* muestra un clúster de Kudu con tres maestros y múltiples servidores de tabletas, cada uno de los cuales sirve a múltiples tabletas.

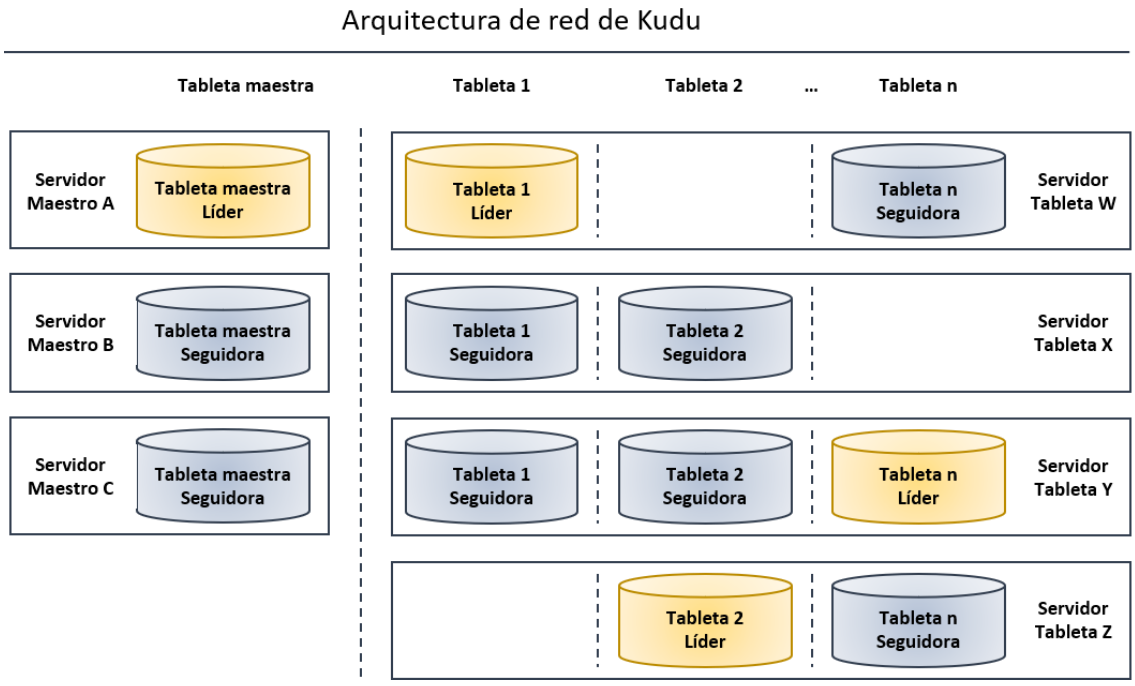


Ilustración 17: Arquitectura Apache Kudu

10.4.3 Comparación

El formato de almacenamiento de Kudu permite actualizaciones de una sola fila, mientras que las actualizaciones de segmentos de Druid existentes requieren volver a crear el segmento, por lo que, en teoría, el proceso para actualizar los valores antiguos debería tener una mayor latencia en Druid. Sin embargo, los requisitos en Kudu para mantener espacio adicional para almacenar actualizaciones, así como para organizar los datos por ID en lugar de por tiempo, tienen el potencial de introducir cierta latencia adicional y el acceso a datos que no son necesarios para responder la consulta en el tiempo de la consulta.

Druid resume/acumula los datos en el momento de la ingestión, lo que en la práctica reduce los datos sin procesar que deben almacenarse significativamente (hasta 40 veces en promedio), y aumenta el rendimiento de escanear los datos sin procesar significativamente. Los segmentos de Druid también contienen índices bitmap para un filtrado rápido, cosa que Kudu actualmente no admite. La arquitectura de segmentos de Druid está muy orientada hacia agregados y filtros rápidos, y para flujos de trabajo OLAP. Los agregados son muy rápidos en Druid, mientras que las actualizaciones de datos más antiguos tienen mayor latencia. Esto es así por el diseño, ya que los datos

con los que Druid es bueno son, generalmente, datos de eventos, y no es necesario actualizarlos con demasiada frecuencia. Kudu admite claves primarias arbitrarias con restricciones de exclusividad y búsqueda eficiente por rangos de esas claves. Kudu elige no incluir el motor de ejecución, pero admite operaciones suficientes para permitir el procesamiento local del nodo desde los motores de ejecución. Esto significa que Kudu puede admitir múltiples marcos en los mismos datos (por ejemplo, MapReduce, Spark y SQL). Druid incluye su propia capa de consulta que le permite bajar agregaciones y cálculos directamente a los procesos de datos para procesamiento de consultas más rápido.

En la *Tabla 14* encontramos las características principales de cada una de ellas para poder compararlas más fácilmente.

	Apache Druid	**Apache Kudu**
Modelo de base de datos	Fragmentación vertical	Fragmentación vertical
Licencia	Libre	Libre
Lenguaje de implementación	Java	C++
Esquema de los datos	Si, aunque soporta columnas sin esquema	Sí
Tipo de arquitectura	Descentralizada	Centralizada
Teorema CAP	AP	CP
Métodos de partición	Sharding	Sharding
Métodos de replicación	Vía HDFS, S3 u otros motores de almacenamiento	Factor de replicación elegible
Compatible con MapReduce	No	Sí
Consistencia	Consistencia fuerte[30]	Consistencia fuerte
Concurrencia	Sí	Sí
Índices secundarios	Sí	No

Tabla 14: Comparativa Druid y Kudu

10.5 Definición de la arquitectura Big Data Warehouse

Tras haber visto todas las herramientas que pueden encajar dentro de cada uno de los componentes de la arquitectura, llega el momento de hacer selección y empezar a ver como estas herramientas se conectan entre sí.

Para el almacén de datos histórico y como hemos podido ver en la comparación realizada sobre las tres herramientas que se han analizado, se optará por HDFS ya que permite la entrada de cualquier tipo de dato (ficheros, tablas, imágenes, …), por su perspectiva en el teorema de CAP hacia la consistencia y la tolerancia de partición y

[30] Con consistencia fuerte, nos referimos a que dan prioridad en el teorema de CAP a la consistencia.

debido a que no requiere un procesado inicial ni cierta estructuración a la llegada de los datos que vendrán en streaming.

En herramientas de procesado de los datos podemos decir que hay un claro vencedor y es que Spark proporciona más velocidad de procesado, proporciona más operaciones para tratar los datos y además se puede programar en varios lenguajes, haciendo la vida más fácil a los desarrolladores. Todos estos beneficios tienen un coste, y es el coste más elevado en hardware, pero se considera que los beneficios compensan ese coste extra.

Para la capa de servicios sí que hay más dudas debido a que son herramientas con enfoques diferentes y potentes en su enfoque, así que aquí lo que decide al final es el aprovechamiento de las otras herramientas que se tienen y se seleccionará Apache Kudu ya que nos otorga gran rapidez en las consultas debido a su particionamiento vertical y porque así se aprovechará Spark para implementar un OLAP por encima.

Por último, para el paso de datos en streaming únicamente se ha valorado una herramienta ya que proporciona lo necesario para dicha arquitectura, Apache Kafka.

Por lo tanto, la arquitectura resultante quedará de la siguiente forma:

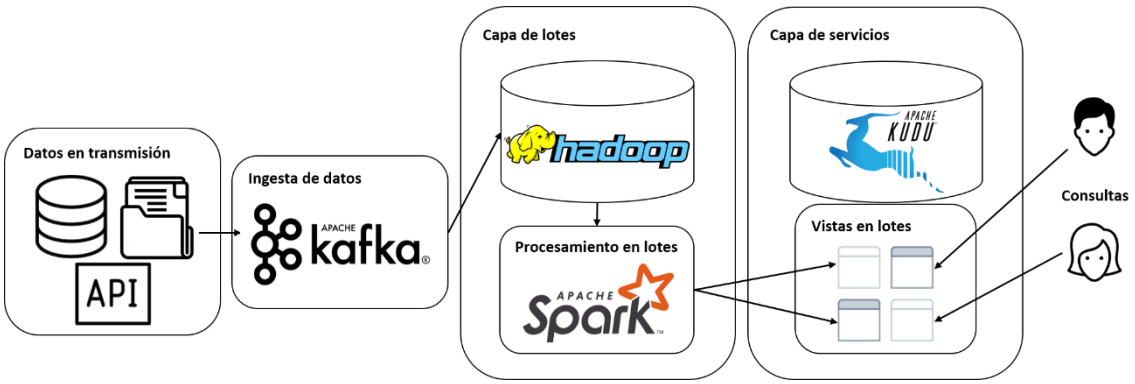


Ilustración 18: Arquitectura resultante con las tecnologías escogidas

Viendo la *Ilustración 18* de la arquitectura resultante, se pueden ver distintas flechas que corresponden a conexiones entre los diferentes componentes de la arquitectura, las cuales se discuten en las siguientes subsecciones.

10.5.1 Conexión de Apache Kafka con Apache HDFS

El conector HDFS permite exportar datos de los temas de Kafka a archivos HDFS en diferentes formatos. El conector sondea periódicamente los datos de Kafka y los escribe en HDFS. Los datos de cada tema Kafka se particionan por el particionador dado y se dividen en fragmentos. Cada fragmento de datos se representa como un archivo HDFS con tema, partición Kafka, inicio y final del offset de este fragmento en el nombre del archivo. Si no se especifica ningún particionador en la configuración, se utiliza el particionador predeterminado que conserva la partición Kafka. El tamaño de cada fragmento de datos está determinado por la cantidad de registros escritos en HDFS, el tiempo escrito en HDFS y la compatibilidad del esquema.

Como característica principal de este conector podemos encontrar que tiene un registro de escritura anticipada para garantizar que cada registro se exporte a HDFS exactamente una vez- Además, el conector administra los offset de los *commits* al

codificar la información del offset de Kafka en el fichero para que podamos comenzar desde los últimos offsets confirmados en caso de fallo o reinicio de tareas.

10.5.2 Conexión de HDFS a Apache Spark

Lo primero que hay que hacer es hacer que las variables HADOOP_CONF_DIR o YARN_CONF_DIR apunten al directorio que contiene los archivos de configuración (del lado del cliente) para el clúster de Hadoop. Estas configuraciones se utilizan para escribir en HDFS y conectarse al YARN ResourceManager. La configuración contenida en este directorio se distribuirá al grupo YARN para que todos los contenedores utilizados por la aplicación utilicen la misma configuración. Si la configuración hace referencia a las propiedades del sistema Java o las variables de entorno no administradas por YARN, también deben configurarse en la configuración de la aplicación Spark (controlador, ejecutores y AM cuando se ejecuta en modo cliente).

Hay dos modos de implementación que pueden usarse para lanzar aplicaciones Spark en YARN. En el modo de clúster, el controlador Spark se ejecuta dentro de un proceso maestro de la aplicación que es administrado por YARN en el clúster y el cliente puede desaparecer después de iniciar la aplicación. En el modo cliente, el controlador se ejecuta en el proceso del cliente y el maestro de la aplicación solo se usa para solicitar recursos de YARN.

A diferencia de otros administradores de clústeres compatibles con Spark en los que la dirección del maestro se especifica en el parámetro --master, en el modo YARN, la dirección del ResourceManager se toma de la configuración de Hadoop, por lo que el parámetro –master es ‘yarn’.

10.5.3 Conexión de Apache Spark a Apache Kudu

Kudu se integra con Spark a través de la API Data Source a partir de la versión 1.0.0

Con esta API se consiguen las siguientes características: soporte para la exploración basada en filas y columnas, poda de columnas y empuje de filtros, puede reportar estadísticas básicas y partición de dato y API de escritura transaccional.

Visto cómo se pueden conectar todos los componentes, lo único que falta es tener una capa OLAP que pueda ser entendible por Kudu y los usuarios puedan realizar sus consultas utilizando la consola o alguna herramienta de visualización tipo Qlik o Tableau, entre otras.

10.5.4 Motor OLAP

Antes de implementar la capa OLAP con Spark, se ve conveniente explicar las diferentes operaciones básicas de OLAP y mostrar algunos ejemplos.

Lo primero que hay que realizar es el cubo básico de los datos que se tenga para poder realizar las diferentes operaciones. En este caso vamos a tener un cubo sencillo en el cuál el hecho va a ser las ventas y van a haber tres dimensiones, producto, ciudad y fecha. El cubo lo podemos ver en la Ilustración 19.

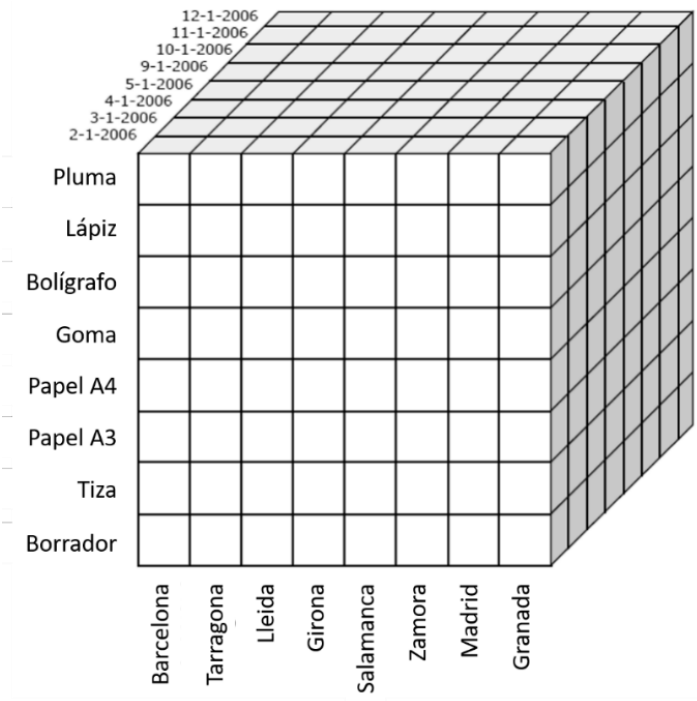


Ilustración 19: Cubo básico OLAP

Una vez se tiene el cubo básico, ya se puede aplicar las diferentes operaciones OLAP, las cuáles se van a explicar en los siguientes párrafos.

Las primeras operaciones OLAP, las operaciones Roll-Up y Drill-Down se van a explicar conjuntamente ya que son operaciones opuestas. En la operación Drill-Down los datos menos detallados se convierten en datos altamente detallados. Se puede hacer tanto bajando en la jerarquía de conceptos como añadiendo una nueva dimensión. En cambio, el Roll-Up realiza la agregación en el cubo OLAP y se puede hacer ascendiendo en la jerarquía de conceptos o reduciendo las dimensiones.

Para que quede más claro, siguiendo el cubo básico de la Ilustración 19, en la Ilustración 20 se van a ver las dos operaciones con un ejemplo.

Ventas	Enero 2006	Febrero 2006	Marzo 2006	Abril 2006
Papel	24	40	15	29
Herramientas de escritura	58	40	59	70

Roll-Up

Drill-Down

Ventas		Enero 2006	Febrero 2006	Marzo 2006	Abril 2006
Papel	Din-A4	24	37	12	27
	Din-A3	0	3	3	2
Herramientas de escritura	Bolígrafo	43	23	36	50
	Lápiz	15	17	23	20

Ilustración 20: Operaciones Roll-Up y Drill-Down

La siguiente es la operación Dice, la cual selecciona un sub-cubo del cubo OLAP seleccionando dos o más dimensiones. De esta definición y fijándonos en el cubo básico de la Ilustración 19, un ejemplo sería escoger ciertos valores de la dimensión de ciudad y de la dimensión de producto, haciendo eso, se cumpliría con la definición que dice que se haga una selección de dos o más dimensiones. Dicho ejemplo, lo podemos ver en la Ilustración 21.

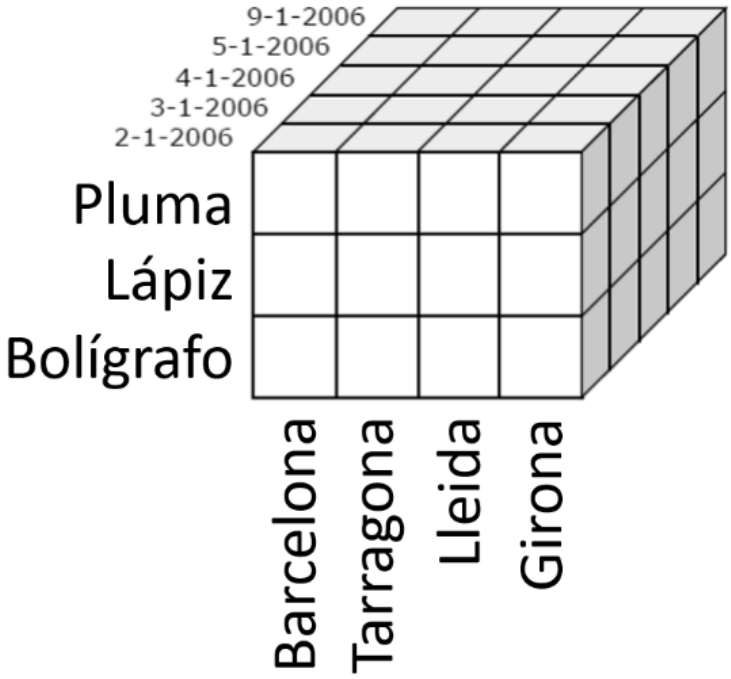


Ilustración 21: Operacón Dice

La cuarta operación es la operación Slice, la cual selecciona una única dimensión del cubo OLAP que da como resultado una nueva creación de sub-cubos. Como dice la definición y basándonos en nuestro ejemplo, hay que escoger una única dimensión, con lo que en la Ilustración 22, en el primer cubo se selecciona de la dimensión producto el producto borrador, en el segundo cubo, escogemos de la dimensión de fecha el dia 2-1-2006 y, en el último cubo, seleccionamos para la dimensión ciudad Barcelona, con lo que se originan tres nuevos sub-cubos.

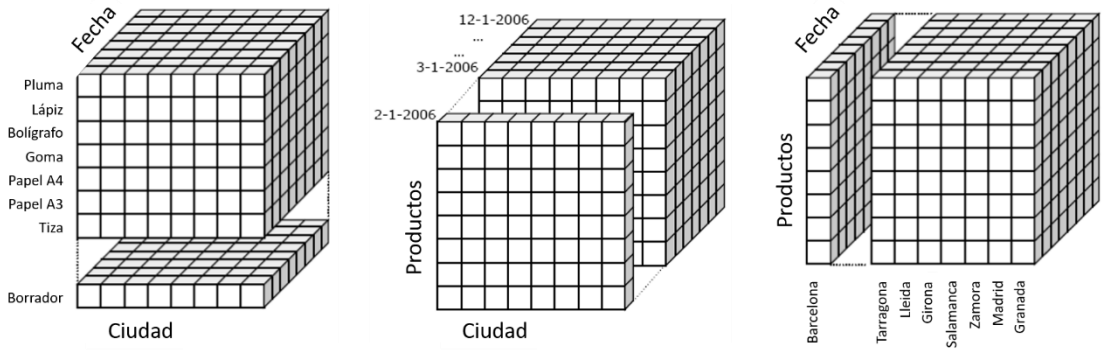
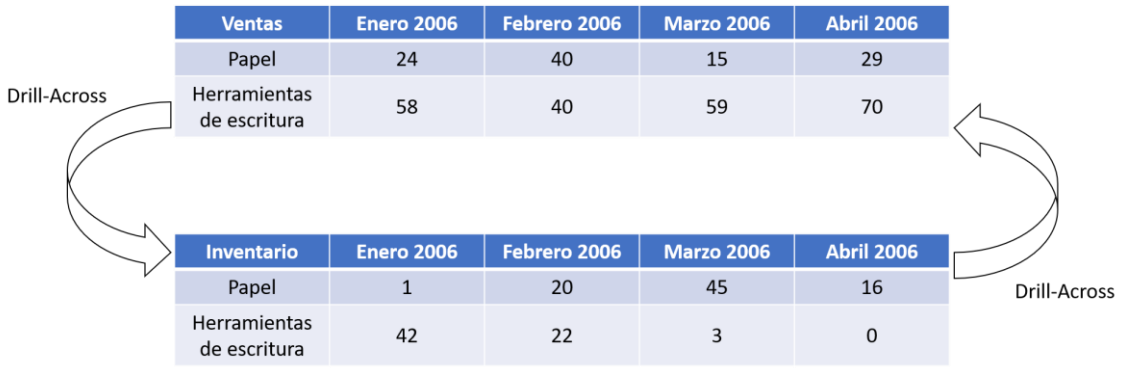


Ilustración 22: Operación Slice

Por último, faltaría la operación Drill-Across, que añade un nuevo tema de análisis a los ya existentes en el cubo. Como podemos ver en el ejemplo de la XX, añadimos a nuestro modelo el hecho inventario, el cual se analiza con las mismas dimensiones que ya teníamos.



Ventas	Enero 2006	Febrero 2006	Marzo 2006	Abril 2006
Papel	24	40	15	29
Herramientas de escritura	58	40	59	70

Inventario	Enero 2006	Febrero 2006	Marzo 2006	Abril 2006
Papel	1	20	45	16
Herramientas de escritura	42	22	3	0

Ilustración 23: Operación Drill-Across

Con todo lo aprendido en esta subsección, se cree que ya se tiene el suficiente conocimiento como para poder implementar el motor OLAP con Spark, lo que se realizará en la siguiente subsección.

10.5.5 Implementación en Spark

Una vez vistas las principales operaciones OLAP, se implementarán en Spark. Primero de todo, vamos a definir un ejemplo de modelado en estrella sencillo para poder definir el cubo básico a partir del cual se realizarán las operaciones OLAP.

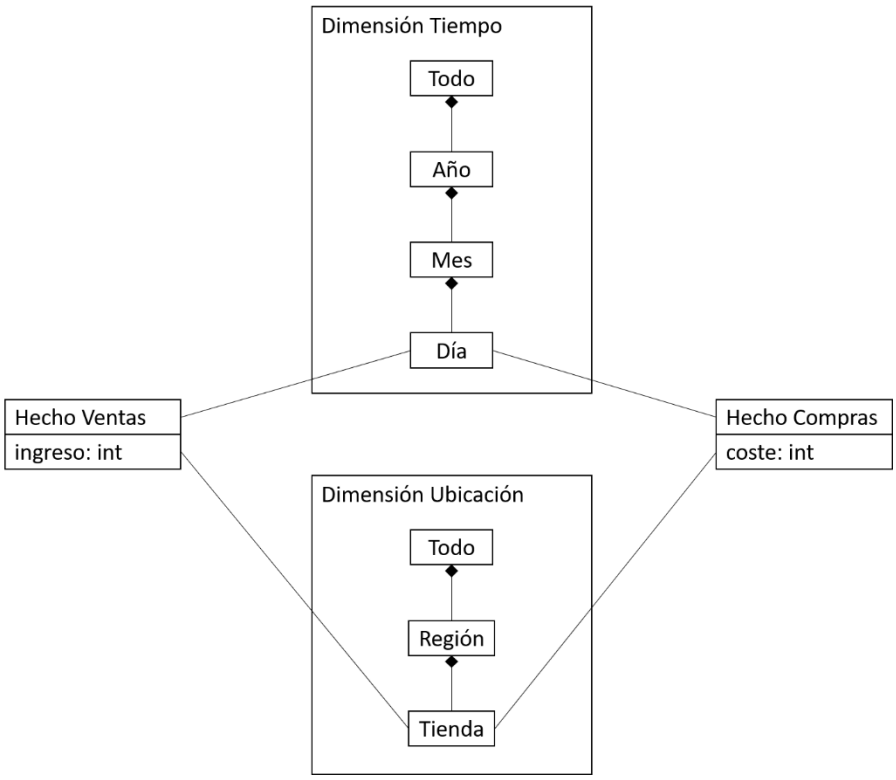


Ilustración 24: Modelo en estrella para OLAP

Lo primero de todo es crear el cubo básico. Para todas las diferentes operaciones que se verán, primero de todo se creará una plantilla que debería seguir cualquier modelo para implementar un OLAP y después se adaptará dicha plantilla al modelo de ejemplo de la Ilustración 24.Además, se enseñarán dichas plantillas en SQL y en Spark.

La consulta SQL del cubo básico sería la siguiente para todo modelo:

```
SELECT dim1.id, dim2.id, …, dimN.id, atrh1, …, atrhN
FROM dim1, …, dimN, hecho
WHERE atrhdim1 = dim1.id AND … AND
        atrhdimN = dimN.id
```

Ilustración 25: Plantilla SQL cubo básico OLAP

Donde básicamente se seleccionarían todos los identificadores de todas las dimensiones que están conectadas (de ahí que miremos si hace *join*) al hecho que se quiere consultar y además también se seleccionan todos los atributos del hecho. La consulta en Spark correspondiente sería la siguiente:

```
hecho.join(dim1, $”id” === $“atrhdim1”)
        ...
        .join(dimN, $”id” === $”atrhdimN”)
        .select(“idDim1”, …, ”idDimN”,”atrh1”, …, “atrhN”)
```

Ilustración 26: Plantilla Spark cubo básico OLAP

Si se adaptan ambas plantillas al modelo de la *Ilustración 24*, se consiguen las siguientes consultas en SQL y Spark para el hecho de Ventas

```
SELECT tienda.id, dia.id, ventas.ingreso
FROM tienda, dia, ventas
WHERE ventas.dia = dia.id AND
        ventas.tienda = tienda.id
```

Ilustración 27: Ejemplo consulta SQL cubo básico

```
ventas.join(tienda, $”id” === $“tienda”)
        .join(dia, $”id” === $”dia”)
        .select(“tienda”, ”dia”, ”ingreso”)
```

Ilustración 28: Ejemplo consulta Spark cubo básico

Para el hecho de Compras sería lo mismo. El siguiente paso sería ver cómo se implementarían las operaciones *Drill-Down* y *Roll-Up.*

En las siguientes consultas veremos la plantilla para hacer un Roll-Up sobre la dimensión 1:

```
SELECT dim1.id, dim1.atrRollUp dim2.id, …, dimN.id, atrh1, …, atrhN
FROM dim1, …, dimN, hecho
WHERE atrhdim1 = dim1.id AND … AND atrhdimN = dimN.id
GROUP BY dim1.atrRollUp
ORDER BY dim1.atrRollUp
```

Ilustración 29: Plantilla SQL operación Roll-Up

```
hecho.join(dim1, $"id" === $"atrhdim1")
        ...
        .join(dimN, $"id" === $"atrhdimN")
        .orderBy("atrRollUpDim1")
        .groupBy("atrRollUpDim1")
        .select("idDim1", …, "idDimN", "atrRollUpDim1","atrh1", …, "atrhN")
```

Ilustración 30: Plantilla Spark operación Roll-Up

En el siguiente ejemplo se va a ver a partir de la consulta de la *Ilustración 27* como hacemos un Roll-Up en la dimensión Ubicación, ya que un Drill-Down en este caso no tendría sentido ya que se está en el nivel más bajo de la jerarquía. Entonces lo que se hace es subir un nivel en la jerarquía de ubicación, por lo tanto, en vez de Tienda se mirará la Región, como se puede ver en las plantillas de la Ilustración 31 y la Ilustración 32:

```
SELECT tienda.id, tienda.region, dia.id, ventas.ingreso
FROM tienda, dia, ventas
WHERE ventas.dia = dia.id AND ventas.tienda = tienda.id
GROUP BY tienda.region
ORDER BY tienda.region
```

Ilustración 31: Ejemplo SQL operación Roll-Up

```
ventas.join(tienda, $"id" === $"idTienda")
        .join(dia, $"id" === $"idDia")
        .orderBy("region")
        .groupBy("region")
        .select("tienda", "dia", "region", "ingreso")
```

Ilustración 32: Ejemplo Spark operación Roll-Up

La siguiente operación será *Dice*. En la plantilla se mostrará dicha operación escogiendo un sub-cubo seleccionando un valor para el identificador de la dimensión 1 y otro para el identificador de la dimensión N, pero se podría hacer con cualquiera siempre que se cojan dos o más.

```
SELECT dim1.id, dim2.id, …, dimN.id, atrh1, …, atrhN
FROM dim1, …, dimN, hecho
WHERE atrhdim1 = dim1.id AND … AND
        atrhdimN = dimN.id AND dim1.id = Valor1
        AND dimN.id = Valor2
```

Ilustración 33: Plantilla SQL operación Dice

```
hecho.join(dim1, $"id" === $"atrhdim1")
        ...
        .join(dimN, $"id" === $"atrhdimN")
        .filter($"idDim1" === Valor1 && $"idDim2" === Valor2)
        .select("idDim1", …, "idDimN","atrh1", …, "atrhN")
```

Ilustración 34: Plantilla Spark operación Dice

En nuestro modelo ejemplo de la Ilustración 24 únicamente tenemos dos dimensiones por lo que escogeremos el valor "tienda1" para tienda y el valor "6" para día. Primero vamos a ver cómo sería esta consulta en SQL y después su transformación a Spark.

```
SELECT tienda.id, dia.id, ventas.ingreso
FROM tienda, dia, ventas
WHERE ventas.dia = dia.id AND
        ventas.tienda = tienda.id AND
        tienda.id = "tienda1" AND
        dia.id = 6
```

Ilustración 35: Ejemplo SQL operación Dice

```
ventas.join(tienda, $"id" === $"tienda")
        .join(dia, $"id" === $"dia")
        .filter($"dia" === 6 && $"tienda" === "tienda1"
        .select("tienda", "dia", "ingreso")
```

Ilustración 36: Ejemplo Spark operación Dice

La siguiente operación es la operación *Slice*. En esta operación, escogeremos únicamente la dimensión de tienda y escogeremos el identificador de la dimensión 1 con "Valor1", con lo que las consultas de plantilla quedarían de la siguiente manera:

```
SELECT dim1.id, dim2.id, …, dimN.id, atrh1, …, atrhN
FROM dim1, …, dimN, hecho
WHERE atrhdim1 = dim1.id AND … AND
        atrhdimN = dimN.id AND dim1.id = Valor1
```

Ilustración 37: Plantilla SQL operación Slice

```
hecho.join(dim1, $"id" === $"atrhdim1")
        ...
        .join(dimN, $"id" === $"atrhdimN")
        .filter($"idDim1" === "Valor1")
        .select("idDim1", …, "idDimN","atrh1", …, "atrhN")
```

Ilustración 38: Plantilla Spark operación Slice

Es muy similar a la operación Dice en cuanto a código, pero únicamente escogiendo una dimensión, por ejemplo la dimensión Tiempo, cogeríamos únicamente el valor día = 6 por lo tanto, las consultas resultantes cogiendo el modelo de ejemplo serían las siguientes:

```
SELECT tienda.id, dia.id, ventas.ingreso
FROM tienda, dia, ventas
WHERE ventas.dia = dia.id AND
        ventas.tienda = tienda.id AND
        dia.id = 6
```

Ilustración 39: Ejemplo SQL operación Slice

```
ventas.join(tienda, $"id" === $"tienda")
        .join(dia, $"id" === $"dia")
        .filter($"dia" === 6)
        .select("tienda", "dia", "ingreso")
```

Ilustración 40: Ejemplo Spark operación Slice

Por último, tenemos la operación *Drill-Across*. En la plantilla, se verá la diferencia con el cubo básico ya que añadiremos un nuevo hecho, con lo que ello conlleva:

```
SELECT dim1.id, dim2.id, …, dimN.id, atrh1, …, atrhN
FROM dim1, …, dimN, hecho1, hecho2
WHERE atrh1dim1 = dim1.id AND … AND
        atrh1dimN = dimN.id AND atrh2dim1 = dim1.id
        AND … AND atrh2dimN = dimN.id
```

Ilustración 41: Plantilla SQL operación Drill-Across

```
aux = hecho1.join(dim1, $"id" === $"atrh1dim1")
        ...
        .join(dimN, $"id" === $"atrh1dimN")
        .select("idDim1", …, "idDimN","atrh1.1", …, "atrh1.N");

Hecho2.join(aux, $"idDim1" === $"atrh2dim1)
        …
        .join(aux, $"idDimN" === $"atrh2dimN)
        .select("idDim1", …, "idDimN", "atrh1.1", …, "atrh1.N",
         "atrh2.1", …, "atrh2.N")
```

Ilustración 42: Plantilla Spark operación Drill-Across

Y para el modelo de ejemplo que se ha diseñado en este proyecto, las operaciones quedarían de la siguiente forma:

```
SELECT tienda.id, dia.id, ventas.ingreso, compras.coste
FROM tienda, dia, ventas, compras
WHERE ventas.dia = dia.id AND ventas.tienda = tienda.id
        AND compras.dia = compras.id AND
        compras.tienda = tienda.id
```

Ilustración 43: Ejemplo SQL operación Drill-Across

```
aux = ventas.join(tienda, $"id" === $"tienda")
        .join(dia, $"id" === $"dia")
        .select("tienda", "dia", "ingreso")

compras.join(aux, $"tienda" === $"idTienda")
        .join(aux, $"dia" === $"idDia")
        .select("tienda", "dia", "ingreso", "coste")
```

Ilustración 44: Ejemplo Spark operación Drill-Across

Con la definición de todas estas operaciones, utilizando las plantillas para cualquier modelo que se quisiera montar dentro de nuestro entorno, podríamos realizar un esquema OLAP.

11 Securización de la arquitectura

Una vez montada la arquitectura, hay que cumplir con las leyes y reglamentos de protección de los datos. Para cumplir con estas leyes y reglamentos existen múltiples opciones para proteger dichos datos, ya sea utilizando herramientas de creación de entornos de prueba, herramientas que descubren datos sensibles o herramientas que sirven para el archivado de datos históricos, entre otras, pero si nos fijamos bien en la base de esas herramientas o en la finalidad u objetivos que tienen podemos ver que todas ellas coinciden en algo, el enmascaramiento. Por ello, en este proyecto se ha decidido realizar un estudio de esta solución para proteger los datos y ver sus diferentes tipos.

Primero de todo, se va a definir que es el enmascaramiento de datos. Todas las plataformas de enmascaramiento de datos reemplazan los elementos con valores similares, y opcionalmente mueven los datos enmascarados a una nueva ubicación. El enmascaramiento crea un sustituto de datos *proxy* que conserva parte del valor del original. El punto es proporcionar datos que se vean y actúen como los datos originales, pero que carezcan de sensibilidad y no representen un riesgo de exposición, lo que reduce el número de controles de seguridad que hay que realizar para repositorios de datos enmascarados. El enmascaramiento debe funcionar con repositorios de datos comunes, como archivos y bases de datos, sin dejar el repositorio inutilizable. La 'máscara' debería hacer imposible o irrealizable revertir los valores enmascarados a los datos originales sin información adicional especial, como un secreto compartido o una clave de cifrado.

Existen cinco leyes que ayudan a la hora de capturar la esencia del enmascaramiento de datos y como éste ayuda en cuanto a seguridad y conformidad. Las leyes son las siguientes:

1. El enmascaramiento no debe ser reversible. Aunque se enmascaren los datos, nunca debería ser posible utilizarlos para recuperar los datos confidenciales originales.

2. Los resultados deben ser representativos de los datos de origen. La razón para enmascarar los datos en lugar de simplemente generar datos aleatorios es proporcionar información no confidencial que aún se asemeje a los datos de producción para fines de desarrollo y test. Esto podría incluir distribuciones geográficas, distribuciones de tarjetas de crédito (tal vez dejando los cuatro primeros dígitos, pero el resto generar números aleatorios), o manteniendo la legibilidad humana de nombres y direcciones (falsos).

3. La integridad referencial debe mantenerse. Las soluciones de enmascaramiento no deben interrumpir la integridad referencial: si el número de una tarjeta de crédito es una clave principal y está codificada como parte del enmascaramiento, todas las instancias de ese número enlazadas a través de pares de claves deben codificarse de manera idéntica.

4. Sólo se enmascararán datos no confidenciales si se pueden usar para recrear datos confidenciales. No es necesario enmascarar toda la base de datos, sólo aquellas partes consideradas confidenciales o sensibles. Pero algunos datos no confidenciales pueden usarse para recrear o vincular datos confidenciales. Por ejemplo, si se codifica una identificación médica pero los códigos de tratamientos para un registro solo se pueden asignar a un registro, también se deben securizar esos códigos.

5. El enmascaramiento debe ser un proceso repetible. El enmascaramiento único no solo es casi imposible de mantener, sino que es bastante ineficaz. Los datos de desarrollo y test deben representar datos de producción en constante cambio lo más iguales posible. Es posible que los datos analíticos deban generarse diariamente o incluso cada hora. Si el enmascaramiento no es un proceso automatizado, es ineficiente, costoso e ineficaz.

11.1 Tipos de enmascaramiento

'Enmascaramiento' es un término genérico que abarca varias variaciones de proceso. En un amplio sentido el enmascaramiento de datos -sólo enmascaramiento para el resto de este documento- abarca la recolección de datos, la ofuscación de datos, almacenamiento de éstos y, con frecuencia, el movimiento de dicha información. Pero enmascarar también se utiliza en referencia a la operación de enmascaramiento en sí misma, es decir, como cambiamos los datos originales a otros similares. Hay muchas formas diferentes de ofuscar datos dependiendo de su tipo, cada uno representado por una función diferente, y cada uno adecuado para diferentes casos de uso. A continuación, se verá una lista de las diferentes maneras de enmascaramiento que se utilizan para ocultar los datos. Estas son las opciones estándar que ofrecen los productos de enmascaramiento del mercado, con el valor particular de cada opción:

- **Sustitución:** es simplemente reemplazar un valor por otro. Por ejemplo, la herramienta podría sustituir los nombres y apellidos de una persona con nombres de una entrada aleatoria de una guía telefónica. Los datos resultantes aún constituyen un nombre, pero no tienen una relación lógica con el nombre real original a menos que tenga acceso a la tabla de sustitución original.

- **Redacción / Anulación:** esta sustitución simplemente reemplaza los datos confidenciales con un valor genérico, como "X". Por ejemplo, se podría reemplazar un número de teléfono por XXX-XXX-XXX o un DNI por XXXXXXXX-X. Esta es la forma más simple y rápida de enmascaramiento, pero la salida proporciona poco o ningún valor.

- **Orden aleatorio (*shuffling*):** es un método para aleatorizar los valores existentes verticalmente en un conjunto de datos. Por ejemplo, barajar valores individuales en una columna de salarios de una tabla de datos de empleados haría que la tabla fuera inútil para aprender lo que cualquier empleado en particular gana sin cambiar los valores agregados o promedio de la tabla. Es una técnica de aleatorización común para desasociar relaciones de datos confidenciales (por ejemplo, Juan gana X € por año) mientras preserva valores agregados.

- **Emborronar (*blurring*):** coger un valor existente y modificarlo para que el valor caiga aleatoriamente dentro de un rango definido.

- **Promediar:** es una técnica de ofuscación en la que los números individuales se reemplazan por un valor aleatorio, pero en todo el campo/columna, el promedio de estos valores permanece constante. En el ejemplo del salario anterior, se podrían sustituir los salarios individuales con el promedio de un grupo o división corporativa para ocultar los valores de los salarios individuales mientras se mantiene una agregación relacionada con los datos reales.

- **Quitar la identidad de los datos (*de-identification*):** un término genérico para cualquier proceso que elimina la información que permite identificar a alguien, como quién ha producido los datos o las identidades personales dentro del conjunto de

datos. Esta técnica es importante para tratar con datos complejos, de varias columnas, que proporcionan pistas suficientes para revertir los datos enmascarados en identidades individuales.

- **Tokenización:** es la sustitución de los elementos con parámetros de sustitución aleatorios, aunque los proveedores abusan del término de tokenización para una variedad de otras técnicas. Los tokens no son reversibles porque el token no tiene relación lógica con el valor original.

- **Encriptación preservando el formato (FPE[31]):** es el proceso de transformación de datos en un estado ilegible. A diferencia de los otros métodos enumerados, el valor original se puede determinar a partir del valor cifrado, pero solo se puede revertir con conocimiento especial (la clave). Mientras que la mayoría de los algoritmos de encriptación producen cadenas de longitud arbitraria, esta técnica transforma los datos en un estado ilegible al tiempo que conserva el formato (apariencia general) de los valores originales. Técnicamente, la encriptación viola la primera ley del enmascaramiento de datos, pero diversos clientes y productos utilizan dicha técnica, por eso se ha añadido.

11.2 Modelos de implementación de enmascaramiento

Vistas las maneras en que se pueden enmascarar los datos, se van a ver cuatro maneras de implementar una solución de enmascaramiento de datos: el enmascaramiento estático, el enmascaramiento físico, el enmascaramiento dinámico basado en vistas y el enmascaramiento dinámico basado en proxy.

11.2.1 Enmascaramiento estático

El enmascaramiento estático también se conoce como ETL ya que trabaja con una exportación desde el repositorio de origen. Cada fase de la ETL se realiza normalmente en servidores separados: un depósito de datos, un servidor de enmascaramiento que organiza la transformación y una base de datos de destino. El servidor de enmascaramiento se conecta al origen, recupera una copia de los datos, aplica el enmascaramiento a columnas de datos específicas y luego carga el resultado en el servidor de destino. Este proceso puede ser completamente automatizado, completamente manual o parcialmente automatizado.

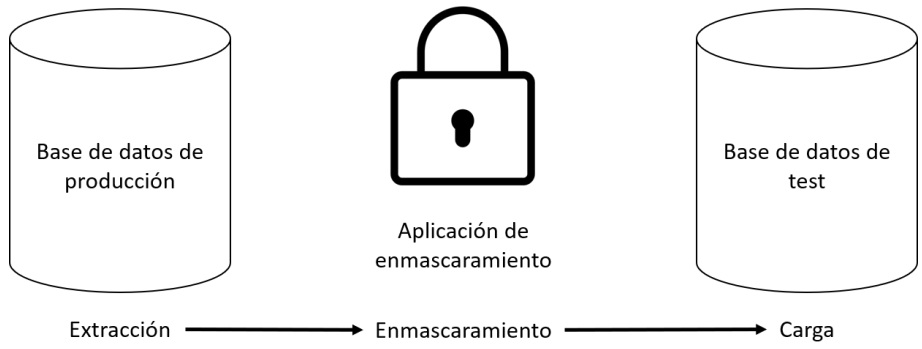


Ilustración 45: Funcionamiento enmascaramiento estático
Inspirado en https://securosis.com/assets/library/reports/UnderstandingMasking_FinalMaster_V3.pdf

[31] Format-Preserving Encryption

Entraremos en más detalle en cada uno de los pasos:

1. **Extracción:** el primer paso es extraer los datos de un repositorio de almacenamiento, la mayoría de las veces, de una base de datos. El proceso de extracción puede seleccionar la base de datos completa o recopilar un subconjunto de los datos en función de algunos criterios de selección. Los datos extraídos a menudo se formatean para facilitar la aplicación del enmascaramiento. Los resultados pueden transmitirse directamente a la aplicación de enmascaramiento para su procesamiento o volcarse en un fichero, como un archivo CSV separado por comas. Los datos extraídos se transfieren de forma segura, como un archivo cifrado o a través de una conexión SSL cifrada, a la plataforma de enmascaramiento.

2. **Transformación:** el segundo paso es aplicar el enmascaramiento a los datos, transformando los datos sensibles de producción en una aproximación segura del contenido original. El enmascaramiento casi siempre se aplica sobre columnas. Por ejemplo, una base de datos puede contener una tabla de clientes, donde cada entrada de cliente incluye su DNI, entonces a esa columna se le aplica el enmascaramiento correspondiente. La aplicación de enmascaramiento analiza los datos y, para cada columna de datos que se va a enmascarar, reemplaza cada entrada en la columna con un valor enmascarado.

3. **Cargar:** en el último paso, los datos enmascarados se cargan en una base de datos destino. Los datos enmascarados se copian en una o más bases de datos destino, donde se vuelven a cargar en las tablas. La base de datos destino no contiene datos confidenciales, por lo que sus requisitos de seguridad y auditoría son más bajos que los de la base de datos original con datos confidenciales.

Este método es el enfoque de enmascaramiento más genérico y flexible. Se implementa en plataformas de enmascaramiento dedicadas, herramientas de base de datos integradas y hasta en scripts de uso doméstico. Debido a que hay muchos entornos diferentes que requieren de enmascaramiento y diferentes requisitos de los clientes, este no es siempre el enfoque correcto, por lo que se verán los enfoques alternativos.

11.2.2 Enmascaramiento físico

En algunos casos, se requiere o se debe crear una copia enmascarada dentro de la base de datos de origen. En algunos casos, se realiza una copa en la base de datos de producción y luego se enmascara, quizás antes de moverse a otra base de datos menos sensible. En otros casos, los datos de producción se mueven sin cambios (de forma segura) a otro sistema, y luego se enmascaran en dicho sistema. Todas estas variaciones se denominan enmascaramiento físico o enmascaramiento en el lugar porque omiten uno o los dos pasos de movimiento de datos que veíamos anteriormente. El enmascaramiento se aplica como antes pero dentro de una base de datos, lo que plantea consideraciones de seguridad y rendimiento.

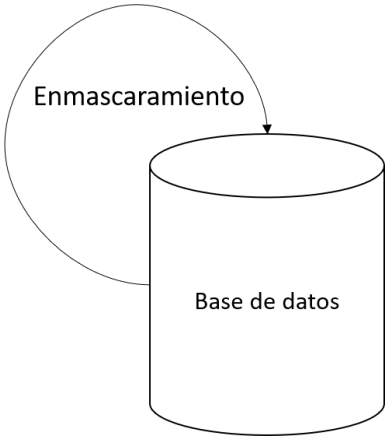


Ilustración 46: Funcionamiento enmascaramiento físico
Inspirado en https://securosis.com/assets/library/reports/UnderstandingMasking_FinalMaster_V3.pdf

Hay muy buenas razones para aplicar este método. La primera es aprovechar la facilidad de las bases de datos con la gestión y la manipulación de los datos. Son perfectas en la transformación de los datos y ofrecen un rendimiento de enmascaramiento muy alto. Aprovechar las funciones incorporadas y los procedimientos almacenados puede acelerar el proceso de enmascaramiento. Enmascarar los datos en su lugar de origen (reemplazar los datos en lugar de crear una nueva copia) protege los archivos de datos y los archivos de bases de datos de la indagación, en caso de que alguien acceda a las cintas de copia de seguridad o los archivos de disco sin procesar.

Si la seguridad de los datos después de salir de la base de datos de producción es una preocupación principal, el anterior modelo y este modelo antes de mover los datos a otra ubicación deben satisfacer los requisitos de seguridad y auditoría. Muchos entornos de prueba tienen poca seguridad, lo que puede hacer que se requiera un enmascaramiento antes de la exportación o un intercambio seguro de ETL para garantizar que los datos confidenciales nunca se expongan en la red o en el repositorio de destino.

Este método no es muy popular debido a la sobrecarga computacional adicional de la operación de enmascaramiento, además de la sobrecarga requerida para leer y escribir los datos que se están transformando, puede tener un impacto inaceptable en el rendimiento de la base de datos. En muchas organizaciones, no se puede absorber dicha carga adicional. El enmascaramiento en la base de datos de destino tampoco es muy popular: las soluciones de enmascaramiento generalmente se adquieren para evitar colocar datos confidenciales en sistemas de prueba inseguros, y los clientes prefieren evitar cargar datos desenmascarados en sistemas de prueba que no son de confianza. También hay que tener cuenta que este tipo de enmascaramiento tiene otros problemas de seguridad: los rastros de los datos originales a menudo se quedan en logs, archivos de transacciones y archivos temporales. Ya sea en la fuente o en la base de datos destino, este residuo puede proporcionar nuevas vías de ataque. Siempre que se realice un enmascaramiento, se tiene que verificar que no quedan copias no deseadas o rastros de datos confidenciales.

Para abordar estos problemas, las soluciones de este tipo se han convertido en una forma de enfoque hibrido, haciendo el enmascarado en los agentes de software locales antes de la inserción en la base de datos, para garantizar que los datos confidenciales nunca lleguen al sistema de prueba. Este enfoque también mitiga los problemas de rendimiento de los clientes y el proceso de enmascaramiento es menos probable que compita con los recursos de la base de datos.

11.2.3 Enmascaramiento dinámico basado en vistas

El enmascaramiento basado en vistas se refiere al almacenamiento de los datos de producción y la versión enmascarada correspondiente en la misma base de datos. La copa de datos que un usuario ve está determinada por la base de datos en el momento de la solicitud a medida que la máscara se activa dinámicamente, según la base de datos, dependiendo de las credenciales del usuario que lo solicita. Cuando un usuario solicita datos, sus credenciales e información de sesión son examinadas por la plataforma de enmascaramiento. Los usuarios autorizados reciben datos de producción desenmascarados, en cambio, los usuarios sin autorización para obtener información confidencial, o que utilizan aplicaciones no autorizadas, o que activan algún otor filtro de seguridad que marca la conexión como sospechosa, reciben datos enmascarados.

El enmascaramiento dinámico basado en vistas se implementa mediante 1) el código de la base de datos para inspeccionar y posiblemente alterar la consulta entrante para seleccionar datos originales o enmascarados, y 2) una "vista" de la base de datos, que no es más que una tabla virtual. Si bien la vista es estructuralmente similar a una tabla real, solo contiene datos enmascarados. Este tipo de enmascaramiento requiere un complemento, un desencadenante o un código que reside en la base de datos para desviar las consultas sospechosas de los datos sin enmascarar.

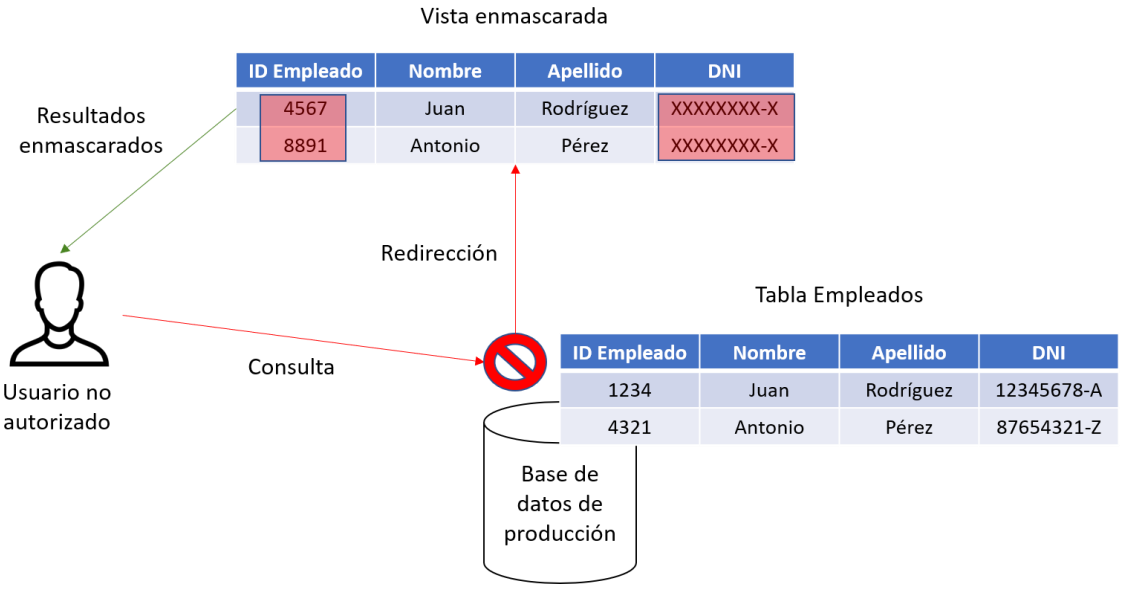


Ilustración 47: Funcionamiento enmascaramiento dinámico basado en vistas
Inspirado en https://securosis.com/assets/library/reports/UnderstandingMasking_FinalMaster_V3.pdf

El enmascaramiento dinámico es una manera de proteger los datos en entornos de aplicaciones web en los que los usuarios no pueden autenticarse perfectamente, sin la costosa recodificación de aplicaciones. Además, el enmascaramiento basado en vistas permite la prueba de aplicaciones de producción en un entorno de producción, al exponer únicamente los datos enmascarados a usuarios de prueba con menos privilegios. Para las empresas con bases de datos de producción muy grandes, que no están preocupadas por una sobrecarga adicional en los servidores de base de datos de producción, este modelo proporciona una excelente aproximación de un entorno de producción para pruebas.

Pero, por supuesto, hay inconvenientes. La mayoría de los clientes no quieren esta carga en los servidores de bases de datos de producción. Además, el enmascaramiento

de la base de datos no evita la fuga de datos, de registros, de archivos de datos o archivos. Esto no es peor que un entorno de producción normal, pero es importante recordar que todos estos otros tipos de datos secundarios son objetivos para los atacantes. Desafortunadamente, este enmascaramiento solo está disponible para bases de datos relacionales, no para sistemas de archivos u otras plataformas. Finalmente, si la vista falla de alguna manera, las consultas no autorizadas pueden exponer datos confidenciales o fallar completamente. El comportamiento de las soluciones de enmascaramiento debe probarse en condiciones adversas.

Pese a que este método sólo esté disponible para bases de datos relacionales, es posible llevar el concepto de este tipo de enmascaramiento al mundo del Big Data, aunque en dicho mundo no sea posible la utilización de vistas. Por ejemplo, en Kudu que es nuestra tecnología, no se dispone de vistas, con lo que es posible crear tablas auxiliares, que funcionarán como vistas, donde se guardarán los datos o un subconjunto de los datos de las tablas que se quieran proteger (una tabla auxiliar por tabla que se quiera enmascarar), pero enmascaradas y después según los permisos de cada usuario, dirigirlo a la tabla con los datos en claro o a la tabla con los datos enmascarados.

11.2.4 Enmascaramiento dinámico basado en proxy

Un modelo de enmascaramiento dinámico de datos más recientes es donde los datos son enmascarados por un servicio en línea a medida que entran o salen de una base de datos. Los resultados de una consulta de un usuario se interceptan y enmascaran sobre la marcha, con resultados enmascarados sustituidos de forma transparente antes de ser devueltos al usuario. Por ejemplo, si un usuario consulta demasiados números de tarjetas de crédito, o si una consulta se origina en una ubicación no aprobada, los datos devueltos pueden ser eliminados. Esto difiere del enmascaramiento basado en vistas en que se produce fuera del repositorio de datos y está disponible para sistemas no relacionales. El proxy puede ser un agente sobre la base de datos, un dispositivo implementado “en línea” entre los usuarios y los datos, para forzar todas las solicitudes a través del proxy. La gran ventaja es que la protección de datos está habilitada sin necesidad de alterar la base de datos; no hay ningún proceso adicional de validación de programación y control de calidad.

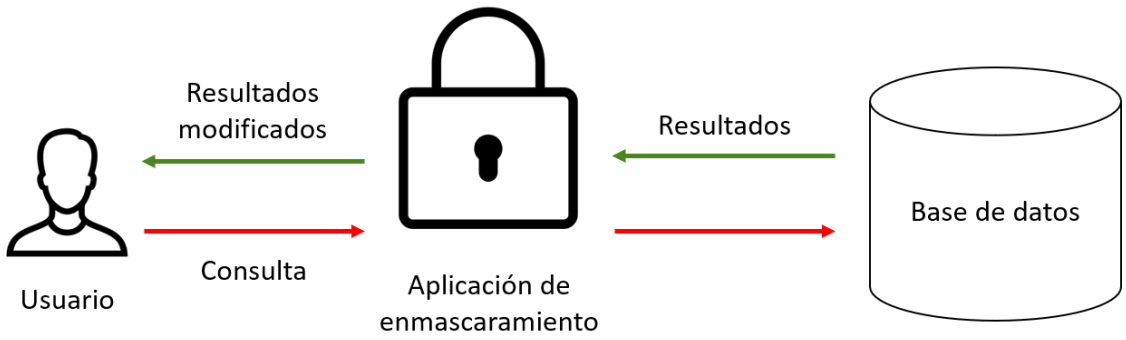


Ilustración 48: Funcionamiento enmascaramiento dinámico basado en proxy
Inspirado en https://securosis.com/assets/library/reports/UnderstandingMasking_FinalMaster_V3.pdf

Otro avance reciente en el enmascaramiento dinámico de datos es la sustitución de consultas. En esta variación, la plataforma de enmascaramiento es lo suficientemente inteligente como para reconocer una solicitud de datos confidenciales. Estas consultas se interceptan y se vuelven a escribir para seleccionar información de una columna diferente (enmascarada). Por ejemplo, una consulta para seleccionar números de

tarjetas de crédito podría modificarse para solicitar valores de token de números de tarjetas de crédito. El modelo de intercepción es muy flexible; la instrucción SELECT puede dirigirse a una "vista" alternativa, a un archivo de valores enmascarados generados previamente, redactar contenido sensible o incluso a hacer join con otra base de datos.

Una desventaja del enmascaramiento dinámico es la necesidad de crear políticas que tengan en cuenta todos los diferentes casos (consultas para aquellos que se sienten cómodos con la terminología de la base de datos) que deberían devolver datos enmascarados. Las herramientas de descubrimiento automatizado y las políticas preconfiguradas son fundamentales para acelerar la implementación y comprender qué datos están en riesgo. Independientemente de lo buena que sea la política pre-creada, se tendrá que invertir tiempo en crear y personalizar políticas para el entorno. Al igual que con cualquier otro servicio en línea, es importante considerar la conmutación por error, el rendimiento y la escalabilidad antes de la implementación de producción.

12 Finalizando el proyecto

Acabada la sección con la explicación del enmascaramiento como solución para proteger los datos, se da por finalizado el desarrollo de este proyecto. Una vez acabado hay que echar una vista atrás y ver que se ha cumplido y que no de lo que se planificó al inicio del proyecto y ver cuáles han sido los motivos. Por ello, se han escogido diversos puntos en que se va a focalizar dicho estudio o vista atrás, que son los que se explicarán a continuación.

12.1 Formulación del problema

El proyecto proporciona una solución a los casos que se planteaban dentro de la formulación del problema, que eran cuando las empresas manejaban grandes cantidades de datos, gran variedad de fuentes de datos, una gran variedad en la estructuración de estos y con una frecuencia de datos muy elevada. La solución dada, a la cual podemos llamar Big DW, cumple con estos requisitos como ya se ha visto en el estudio, y el proyecto proporciona toda la información suficiente para que las empresas puedan montar sus arquitecturas para poder resolver este problema.

12.2 Objetivos y alcance

En la planificación inicial se planteaban unos objetivos y, a partir de estos objetivos, se planteaba el alcance del proyecto, por lo que estos dos puntos los podemos concluir conjuntamente. En el proyecto se tenía un objetivo principal del cual colgaban otros cuatro objetivos.

El objetivo principal, orquestar los elementos de una arquitectura Lambda para llevar el concepto de DW, se ha cumplido con el desarrollo de la *sección 10*, viendo los distintos componentes que componen dicha arquitectura y qué tecnologías se pueden utilizar en cada uno de ellos. En este objetivo, sí que se ha concretado un poco más el alcance ya que únicamente se han estudiado como máximo dos tecnologías por componente (en algunos solo uno porque era la tecnología líder indiscutible en el mercado o las alternativas no acababan de proporcionar lo que se necesitaba). Se ha tenido que poner un límite debido a que tecnologías podríamos encontrar infinitas y los plazos son los que son y se debía acotar el tiempo.

El primero de los cuatro objetivos que colgaban del objetivo principal, estudiar las arquitecturas tradicionales de DW, se realizó en la *sección 7*, estudiando las arquitecturas propuestas por Ralph Kimball y W.H. Inmon, como ya se había planteado en el alcance del proyecto.

El segundo, ver las limitaciones de dichas arquitecturas cuando se les somete a ciertos requisitos, se puede ver en la *sección 8*, donde se hace un estudio para cada uno de esos requisitos.

El tercero era estudiar las arquitecturas Big Data, que se puede encontrar en la *sección 9* y se centra en las arquitecturas Lambda y Kappa, tal y como se había definido en el alcance.

Por último, se tenía como objetivo aplicar seguridad en la arquitectura final, el cual se ha explicado en la *sección* 11, explicando las maneras de aplicar seguridad en los datos mediante el uso del enmascarado.

12.3 Planificación del proyecto

Las tareas que se definieron durante la fase de planificación del proyecto se han mantenido tal cual se describieron en la *sección 5.1* exceptuando las tareas de orquestación de las arquitecturas Big Data y la securización de éstas. En la orquestación de las arquitecturas Big Data, tal y como se ha dicho en la sección anterior, se ha acotado el número de tecnologías que se iban a estudiar con el propósito de realizar la fase en los tiempos estimados, además de que la comparación de las herramientas ha llevado más tiempo del estimado inicialmente, debido a que había que ver sobre qué puntos compararlas. En la securización de los datos, se ha optado por explicar a fondo una de las técnicas más utilizadas para proteger los datos como es el enmascaramiento. Lo que no se ha seguido con la planificación ha sido realizar un estudio de las tecnologías de enmascaramiento que hay en el mercado básicamente por dos motivos: 1) la información que se puede encontrar de cada uno de los productos en internet, ya sea en su propia página web o en otras no es suficiente para llegar a un análisis profundo de la herramienta y así realizar una comparación entre ellas y, 2) las relaciones entre empresas siempre son lentas ya que hay acuerdos legales e intereses de por medio y, con la relación de everis y las empresas que poseen dichos productos también ha sucedido. Dentro de everis se estaba realizando una recogida de funcionalidades y una comparativa entre las diferentes tecnologías que se habían detectado, pero para completar dicha recogida se necesitaba una demo o documentación mucho más extensa como un manual, cosa que únicamente se ha obtenido de una tecnología con lo cual se ha descartado dicho punto y se ha profundizado más en la técnica del enmascaramiento explicando las formas de aplicarlo y las reglas/máscaras que se pueden llegar a utilizar.

Por suerte, no han aparecido los obstáculos que se habían detectado que podían surgir durante el proyecto y, por lo tanto, se ha podido cumplir con los diferentes plazos que se habían planificado, manteniendo el colchón de horas que se había proporcionado para posibles imprevistos.

Dentro de la planificación inicial, no se había tenido en cuenta la fase de conclusiones del proyecto una vez acabado el desarrollo de éste por lo que esas horas extras se han utilizado para realizar las diferentes conclusiones del proyecto.

Con todo esto, el cálculo del tiempo utilizado es el siguiente, que es un poco diferente al realizado en el tiempo estimado de la *sección 5.3*:

Tarea	Código de tarea	Tiempo real (h)	Dependencias
Planificación del proyecto	PPO	76	
Definición del alcance	PPO-A	38	
Gestión del proyecto	PPO-G	38	PPO-A
Estudio de las arquitecturas tradicionales de DW	ADW	52	PPO
Definición de DW	ADW-D	10	
Estudio de la arquitectura Kimball	ADW-K	22	
Estudio de la arquitectura Inmon	ADW-I	20	
Estudio de las limitaciones de las arquitecturas tradicionales	LAT	28	ADW-K, ADW-I
Estudio de las arquitecturas Big Data	ABD	52	
Definición del Big Data	ABD-D	10	
Estudio de la arquitectura Lambda	ABD-L	22	
Estudio de la arquitectura Kappa	ABD-K	20	
Orquestación de las arquitecturas Big Data	OBD	136	ADW,LAT,ABD
Estudio de los componentes	OBD-C	46	
Estudio de las tecnologías	OBD-T	46	
Comparativa de las tecnologías	OBD-TC	44	
Securización de las arquitecturas Big Data	SBD	83	
Estudio del enmascaramiento	SBD-T	40	
Estudio de las maneras de aplicarlo	SBD-M	43	
Finalización del proyecto	FPO	30	SBD
Comparación con planificación inicial	FPO-PI	22	
Conclusiones	FPO-C	8	
Revisión del proyecto	RPO	20	FPO
Total		**477**	

Tabla 15: Tiempo real de cada tarea

Comparando esta tabla con la *Tabla 1* de la *sección 5.3* podremos ver las diferencias que ya se han comentado justo en esta sección. Una vez vista la tabla de tiempos reales, es decir, la *Tabla 15*, el diagrama de Gantt final de este proyecto quedaría de la siguiente forma:

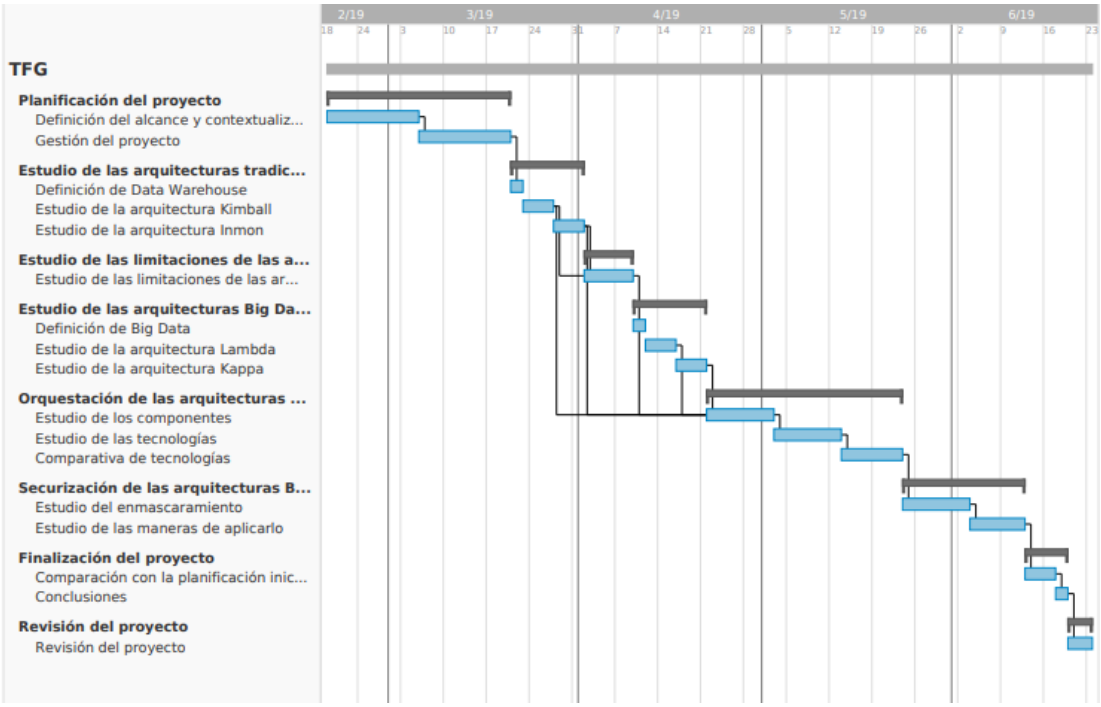


Ilustración 49: Diagrama de Gantt Final
Generado con https://teamgantt.com

12.4 Presupuesto

Debido al cambio de la planificación y también al uso de nuevos recursos durante el desarrollo del proyecto, el presupuesto de éste también ha variado y es lo que se va a ver dentro de esta sección. Se seguirá el orden seguido en la estimación del presupuesto en la *sección 6*.

El presupuesto en hardware no ha variado respecto a lo que se planificó en la fase inicial, con lo que el valor que podemos ver en la *Tabla 2* es el valor final.

El presupuesto de software sí que ha variado respecto a la fase inicial ya que se ha hecho uso de la plataforma SafariBooks con su suscripción y de un *report* de Gartner para poder utilizar los recursos e información que contienen, como por ejemplo los libros donde Ralph Kimball y W.H. Inmon definen sus arquitecturas o una *overview* general sobre el mundo del Big Data, entre otros. Por lo tanto, la nueva tabla de costes de software queda de la siguiente manera:

Producto	Precio	Vida útil	Amortización
Windows 10 Home	145 €	3 años	18,13 €
macOS Sierra	21,99 €	3 años	2,75 €
Microsoft Office	69 €	3 años	8,63 €
TeamGantt	0 €	3 años	0 €
Mendeley Desktop	0 €	3 años	0 €
Trello	0 €	3 años	0 €
Suscripción SafariBooks	154,81 €	3 años	19,35 €
Informes de Gartner	436,64 €	3 años	54,58 €
Total	**827,44 €**		**103,44 €**

Tabla 16: Coste de software final

Si se mira el presupuesto de recursos humanos de la *subsección 6.1.3* se verá que fue calculado a partir de la planificación inicial del tiempo que se hizo, con lo que si ha cambiado dicha distribución de tiempo, como podemos ver en la sección anterior, entonces el presupuesto de recursos humanos también debe cambiar.

Como se hizo en la *subsección 6.1.3*, veremos un desglose por roles y otro por tarea y rol, para que la comparación entre la estimación inicial y la final sea más fácil de realizar.

Rol	Precio por hora	Horas	Salario	Seguridad Social (34%)	Precio final
Director	23,67 €	18	426,06 €	144,86 €	570,92 €
Jefe de proyecto	18,93 €	132	2.498,76 €	849,58 €	3.348,834 €
Desarrollador	11,36 €	327	3.714,72 €	1.263,00 €	4.977,72 €
Total		**477**	**6.639,54 €**		**8.896,98 €**

Tabla 17: Coste de recursos humanos

Tarea	Duración (h)	Dedicación (h)			Coste
		Director	Jefe de proyecto	Desarrollador	
PPO	76	2	66	8	1.387,60 €
PPO-A	38	1	35	2	708,94 €
PPO-G	38	1	31	6	678,66 €
ADW	52	3	5	44	665,50 €
ADW-D	10	1	1	8	133,48 €
ADW-K	22	1	2	19	277,37 €
ADW-I	20	1	2	17	254,65 €
LAT	28	1	5	22	368,24 €
ABD	52	3	5	44	665,50 €
ABD-D	10	1	1	8	133,48 €
ABD-L	22	1	2	19	277,37 €
ABD-K	20	1	2	17	254,65 €
OBD	136	3	12	121	1.672,73 €
OBD-C	46	1	4	41	565,15 €
OBD-T	46	1	4	41	565,15 €
OBD-TC	44	1	4	39	542,43 €
SBD	83	2	8	73	1.028,06 €
SBD-T	40	1	4	35	496,99 €
SBD-M	43	1	4	38	531,07 €
FPO	30	3	21	6	536,70 €
FPO-PI	22	1	19	2	406,06 €
FPO-C	8	2	2	4	130,64 €
RPO	20	1	10	**9**	326,57 €
Total	**477**	**18**	**132**	**327**	**6.639,54 €**

Tabla 18: Coste y tiempo real por tarea

Los costes indirectos han variado respecto a la fase inicial ya que en esta se calculó el caso peor y se han conseguido minimizar gastos.

Producto	Precio	Unidades	Coste estimado
Consumo ordenador	0.133 €/ kWh	0,238 kWh * 160h	5,06 €
Fibra	34,91 €	4,5 meses	157,10 €
Luz	0.133 € / kWh	0,240 kWh * 120h	3,83 €
Alquiler	95 € / mes	4,5 meses	427,5 €
Total			**593,49 €**

Tabla 19: Costes indirectos reales

Por suerte, no han ocurrido desviaciones graves por lo que el presupuesto de costes imprevistos y el de contingencia no se ha tenido que utilizar. Sabiendo esto y habiendo visto las diferencias con la planificación inicial, el coste total del proyecto es el siguiente:

Concepto	Coste estimado
Recursos Hardware	1.800,00 €
Recursos Software	827,44 €
Recursos Humanos	8.896,98 €
Recursos indirectos	593,49 €
Total	**12.117,91 €**

Tabla 20: Coste total del proyecto

Como podemos ver, en comparación al presupuesto estimado (*subsección 6.1.6*) ha habido una desviación del 0,64% por lo que se podría decir que se realizó una buena estimación.

12.5 Metodología y rigor

Durante el proyecto, en cada una de las fases que se habían marcado, se ha seguido una metodología ágil, como ya se había planeado, en la cuál había para cada fase, cuatro sub-fases: una de análisis, una de diseño, otra de desarrollo y otra de revisión. Durante la fase de análisis, se estudiaba el tema objeto de la sección que se quería realizar, una vez detectadas todas aquellas fuentes de información que podían tener interés para el proyecto se entraba en la fase de diseño en la cual se organizaban los puntos de los cuáles se quería hablar y qué longitud iban a tener y, a partir de ese esquema, se entraba en la fase de desarrollo. Por último, se realizaba una revisión individual, en la cual se miraba la planificación inicial mirando si se habían tocado todos los puntos que se habían planificado, si se había cumplido con los plazos y los recursos que se habían utilizado. También se enviaba el documento al equipo dentro de everis para su revisión y al ponente, se recogían los cambios/sugerencias que exponían y se iban aplicando.

Durante el desarrollo se han utilizado Trello y la herramienta de revisión de Microsoft Word como apoyo para facilitar sobre todo la etapa de revisión.

12.6 Sostenibilidad y compromiso social

12.6.1 Impacto ambiental

Para cuantificar el impacto ambiental del proyecto, se utilizará la *Tabla 9* de la *subsección 6.3.1*, aunque no serán exactamente los mismos valores ya que al cambiar las horas dedicadas, el consumo energético también variará. Teniendo en cuenta los recursos del proyecto que consumen energía, el consumo energético del proyecto es el siguiente:

Recurso	Consumo	Horas	Consumo total
MacBook Pro	238 W	160	38,08 kWH
Director	1,6 W	18	0,03 kWH
Jefe de proyecto	1,6 W	132	0,20 kWH
Desarrollador	1,6 W	327	0,53 kWH
Luz	240 W	120	28,80 kWH
Total			**67,64 kWH**

Tabla 21: Consumo energético

Finalmente, el coste energético se ha reducido un 69,34% respecto a la estimación que se realizó durante la fase de planificación. Es un número considerable, aunque hay que tener en cuenta que dicha estimación se hizo considerando el caso peor, por lo que era sencillo mejorar dicho número aplicando las medidas que ya se habían mencionado en esa subsección, como realizar el proyecto durante horas de luz o no tener el ordenador siempre conectado a la corriente.

Este número podría haber mejorado aún más sobre todo mejorando el aspecto de la luz, ya que el aspecto del ordenador es difícil ya de mejorar ya que se aprovechaba hasta que quedase un 5% de batería y quitándolo de corriente cuando llegaba a 100% aunque se podrían haber mejorado los límites, pero tampoco se quería perder el trabajo realizado por no darse cuenta. En el tema luz se podría haber mejorado escogiendo una franja horaria en la cual todo fuesen horas de luz natural, pero eso ha sido imposible para el desarrollador ya que el proyecto se tenía que realizar en franjas de 16 a 21 con lo que siempre había un par de horas en que se necesitaba luz artificial.

Durante la vida útil de este proyecto, los recursos que se pueden gastar o usar para utilizarlo es papel y el consumo energético de la impresora para imprimirlo. Esto se reduce utilizando la versión PDF viéndola tanto online como en local, con lo que no hay la necesidad de realizar ese gasto de recursos naturales. También podría ser que se quisiera realizar una nueva versión del documento, lo que aumentaría el uso de recursos.

12.6.2 Impacto económico

En la *sección 12.4* se ha proporcionado una descripción detallada de los costes que ha tenido el proyecto, teniendo en cuenta tanto recursos materiales como humanos.

Si se compara con la estimación de la fase inicial el presupuesto ha aumentado un 0,64% por lo que no se han podido reducir los costes finales. Se ha reducido el coste energético, pero ha aumentado el coste de software, pero se cree que está justificado ya que se han conseguido fuentes de información con más calidad, y, por lo tanto, la utilidad del proyecto al final será mejor ya que tendrá información más contrastada. Como ya se mencionó en la fase inicial había posibilidades de ahorro con software libre o un equipo más barato, pero no había margen de maniobra ya que tanto el software como el hardware ya se disponía antes de la ejecución del proyecto, por lo que no se ha elegido.

El proyecto durante su vida útil puede tener diferentes costes. El primero serían costes de impresión si se quisiese imprimir, ya que habría que disponer de una impresora, el consumo de ésta y el papel, aunque como se ha mencionado en el apartado ambiental no es una buena opción si se quiere reducir la huella ecológica. Otro coste que podría tener es si se quiere realizar una nueva versión del documento, lo cual haría tener que planificar esa nueva versión, pagar a las personas que participen en el desarrollo de esa nueva versión y pagar los recursos que utilicen en el desarrollo del proyecto. En este último caso, se podrían reducir costes mirando la planificación y el resultado de esta versión y evitar cometer los mismos errores para así ahorrar lo máximo posible.

12.6.3 Impacto social

La realización de este proyecto ha servido para que el desarrollador y su equipo puedan conocer más a fondo las arquitecturas que se estudian, así como las diferentes tecnologías utilizadas. También servirá para ayudar a las empresas a dar el paso al mundo Big Data manteniendo la estructura o el concepto de las arquitecturas de DW que tienen actualmente.

Siempre que se habla de almacenamiento de datos hay que tener en cuenta que los datos, al fin y al cabo, son poder para las diferentes empresas que lo poseen. Por eso, es importante el uso que las empresas hacen de esos datos, ya que pueden manipular a la sociedad gracias a saber cierta información de las personas. Por ejemplo, a una persona que vean que realiza muchas compras, es decir, que sean compradoras compulsivas, las empresas pueden intentar aprovecharse de ellas y enviar ofertas más a menudo, para que caigan y compren, y aquí ya entraríamos en temas de ética que se están intentando reglar con los nuevas leyes y reglamentos de protección de datos. Por ello, la solución que se estudia en este proyecto para proteger los datos, el enmascaramiento, es importante para que las empresas no abusen de poder con los datos que poseen, ya que, al protegerlos, se está “borrando” la personalidad o la identidad de los datos y con ello, se conseguirá que no puedan detectar, siguiendo el ejemplo mencionado antes, a esas personas que tienen problemas con las compras.

Siguiendo esta línea, con la protección de los datos se consigue que con cualquier fuga o robo de información se pierda valor en los datos, ya que los datos que se roben irán enmascarados y, por tanto, no se podrán aprovechar y utilizarlos en contra de las empresas o los clientes de éstas.

13 Conclusiones

Con la finalización de este proyecto, se puede ver que se puede simular el comportamiento de un DW en una arquitectura Big Data y con ello solventar los problemas que tienen los DW tradicionales con ciertos requisitos. Con este proyecto se tiene la base para poder montar una arquitectura y se ofrecen ciertas tecnologías que se desenvuelven bien dentro de ese entorno, así como las plantillas para disponer de un esquema OLAP con Spark.

Cabe decir también que estas arquitecturas propuestas sirven para requisitos de grandes volúmenes de datos, con gran variedad de fuentes y diversas latencias con lo que servirá si eso es lo que principalmente se está buscando como sistema. Con las tecnologías escogidas pasa lo mismo, por ejemplo, las bases de datos columnares están escogidas para que las consultas sean muy rápidas, que es lo que se quería, pero probablemente para muchos otros problemas no sean la solución más óptima y haya que escoger otras tecnologías que se amolden más.

Una vez escogida la arquitectura, se ha visto que la seguridad de los datos es un tema muy importante, y, siempre que se tengan datos sensibles, hay que tenerla. Como se ha visto existen múltiples opciones de colocar una máscara a los datos y varias maneras de implementar una solución de enmascaramiento, que al final, se van a escoger dependiendo de los intereses y los requisitos de cada una de las empresas.

13.1 Competencias técnicas

Al final en este proyecto se han trabajado diferentes ámbitos y se van a intentar resumir en esta sección, intentando clasificar las diferentes secciones del proyecto.

Las secciones que van de la 1 a la 6 y la *sección 12* están dedicadas a la gestión del proyecto y a identificar, evaluar y gestionar los riesgos potenciales asociados al desarrollo de este proyecto, con sus propias medidas en caso de que los riesgos surjan y un plan de acción para solventarlos.

Se podría decir que las demás secciones van más a la parte del tema del proyecto en sí, donde hemos visto el diseño y la implementación (teórico) de un DW, pero con tecnologías Big Data y como se conectan entre sí. Estas tecnologías comparten una característica y es que son aplicaciones distribuidas con lo que trabajan en diferentes nodos y se comunican a través de la red, ya sea local o no. Además, también al ser algunas de estas tecnologías bases de datos, se ha visto un poco la administración de ellas y como especificar, diseñar (como ya se vio en el diseño del OLAP), implementar y evaluar bases de datos, sobre todo en la *sección 10*. En la *sección 8*, viendo las limitaciones de las arquitecturas de DW tradicionales, a su vez, y como ya se había hecho en otras secciones, pero en menos detalles, se estaban definiendo los requisitos que tenía que cumplir la nueva arquitectura de Big DW.

14 Futuros pasos

La realización de este proyecto se ha hecho de manera teórica debido a que el tiempo que se tenía era limitado y se querían tocar diversos temas para definir la nueva arquitectura, con lo que en el futuro sería interesante utilizando este documento implementar la arquitectura que se ha definido en el documento. Una vez montada, se le sometería a los requisitos que se habían definido para ver si realmente el rendimiento es bueno e incluso se podría comparar la nueva arquitectura una vez montada con las arquitecturas base en las cuáles nos hemos basado para ver si el rendimiento es superior mirando diferentes casos de uso.

Así que ese sería el paso siguiente principal, realizar en varios nodos ya sea en el *cloud* o en local la instalación de las diferentes tecnologías, configurarlas y conectarlas entre sí para que exista un flujo de datos, coger un conjunto de datos que pueda ser útil y contenga posibles datos sensibles y crear el OLAP sobre ese conjunto de datos. Una vez montada la arquitectura, se utilizaría alguna de las herramientas de enmascaramiento que se han analizado dentro de nuestro departamento y se instalaría en el entorno para ver si se pueden proteger los datos y los efectos que tiene la herramienta sobre el entorno (si produce un tiempo extra en las consultas, afecta el rendimiento, …).

15 Referencias

[1] home | everis Global. Recuperado el 19 de febrero de 2019 de https://www.everis.com/global/es

[2] Business Intelligence - BI - Gartner IT Glossary. Recuperado el 19 de febrero de 2019 de https://www.gartner.com/it-glossary/business-intelligence-bi/

[3] Data Warehouse | Gartner. (2018). Recuperado el 19 de febrero de 2019 de https://www.gartner.com/it-glossary/data-warehouse/

[4] What Is Big Data? - Gartner IT Glossary - Big Data. Recuperado el 19 de febrero de 2019, de https://www.gartner.com/it-glossary/big-data/

[5] NoSQL Relational Database Management System: Home Page. (2007). Recuperado de http://www.strozzi.it/cgi-bin/CSA/tw7/I/en_US/nosql/Home Page

[6] Columbus, L. Big Data Analytics Adoption Soared In The Enterprise In 2018. Recuperado el 19 de febrero de 2019, de https://www.forbes.com/sites/louiscolumbus/2018/12/23/big-data-analytics-adoption-soared-in-the-enterprise-in-2018/#552f81f0332f

[7] Kimball, R., & Ross, M. (2013). The data warehouse toolkit: the definitive guide to dimensional modeling. Recuperado de https://www.worldcat.org/title/data-warehouse-toolkit-the-definitive-guide-to-dimensional-modeling/oclc/1014070433&referer=brief_results

[8] Inmon, W. H. (2005). Building the data warehouse (4th ed.). Recuperado de https://www.worldcat.org/title/building-the-data-warehouse/oclc/780855910&referer=brief_results

[9] Verrilli, M. (2017). From Lambda to Kappa: A Guide on Real-time Big Data Architectures. Recuperado el 20 de febrero de 2019, de https://www.talend.com/blog/2017/08/28/lambda-kappa-real-time-big-data-architectures/

[10] Druid | Interactive Analytics at Scale. Recuperado el 20 de febrero de 2019, de http://druid.io/

[11] EUGDPR – Information Portal. Recuperado el 20 de febrero de 2019, de https://eugdpr.org/

[12] Microsoft Word: software de procesamiento de textos | Office. Recuperado el 20 de febrero de 2019, de https://products.office.com/es-es/word

[13] Google Drive: almacenamiento en la nube, copias de seguridad de fotos, documentos y mucho más. Recuperado el 20 de febrero de 2019, de https://www.google.com/intl/es_ALL/drive/

[14] Herramientas de colaboración y software de chat de grupo: Microsoft Teams. Recuperado el 20 de febrero de 2019, de https://products.office.com/es-es/microsoft-teams/group-chat-software

[15] Realizar un seguimiento de los cambios en Word - Word. Recuperado el 20 de febrero de 2019, de https://support.office.com/es-es/article/realizar-un-seguimiento-de-los-cambios-en-word-197ba630-0f5f-4a8e-9a77-3712475e806a#ID0EAACAAA=Versiones_más_recientes

[16] *TECNOLOGÍA TENDENCIAS DEL MERCADO LABORAL*. Recuperado el 19 de marzo de 2019, de https://www.michaelpage.es/sites/michaelpage.es/files/PG_ER_IT.pdf

[17] Tabla de tipos de interés, activos y pasivos, aplicados por las entidades de crédito. Recuperado el 19 de marzo de 2019, de https://clientebancario.bde.es/pcb/es/menu-horizontal/productosservici/relacionados/tiposinteres/guia-textual/tiposinteresprac/Tabla_de_tipos__a0b053c69a40f51.html

[18] Ponniah, P. (2010). Data Warehousing Fundamentals for its Professionals. https://doi.org/10.1002/9780470604137

[19] Todo lo que querías saber sobre DataWarehouse (I) - Adictos al trabajo. (s. f.). Recuperado 26 de marzo de 2019, de https://www.adictosaltrabajo.com/2007/10/30/datawarehouse/

[20] Of Data Warehouses, Operational Data Stores, Data Marts and Data «Outhouses». (s. f.). Recuperado 7 de abril de 2019, de https://www.gartner.com/doc/487764/data-warehouses-operational-data-stores

[21] Abramson - Inmon vs Kimball | Data Warehouse | Computing. (s. f.). Recuperado 7 de abril de 2019, de https://es.scribd.com/document/30109013/Abramson-Inmon-vs-Kimball

[22] Las 7 V del Big data: Características más importantes - IIC. (s. f.). Recuperado 13 de abril de 2019, de http://www.iic.uam.es/innovacion/big-data-caracteristicas-mas-importantes-7-v/

[23] Marz, N., & Warren, J. (James O.) (2015). Big data: principles and best practices of scalable real-time data systems. Recuperado de https://www.worldcat.org/title/big-data-principles-and-best-practices-of-scalable-real-time-data-systems/oclc/927375328&referer=brief_results

[24] Big Data: ¿En qué consiste? Su importancia, desafíos y gobernabilidad. (s. f.). Recuperado 13 de abril de 2019, de https://www.powerdata.es/big-data

[25] ¿Qué es Big data? - Definición en WhatIs.com. (s. f.). Recuperado 13 de abril de 2019, de https://searchdatacenter.techtarget.com/es/definicion/Big-data

[26] Qué es Big Data | Universidad Complutense de Madrid. (s. f.). Recuperado 13 de abril de 2019, de https://www.masterbigdataucm.com/que-es-big-data/

[27] Lambda Architecture » λ lambda-architecture.net. (s. f.). Recuperado 20 de abril de 2019, de http://lambda-architecture.net/

[28] The Best Data Processing Architectures: Lambda vs Kappa. (s. f.). Recuperado 20 de abril de 2019, de https://datashark.academy/the-best-data-processing-architectures-lambda-vs-kappa/

[29] James Kinley (The Lambda architecture: principles for...). (s. f.). Recuperado 20 de abril de 2019, de http://jameskinley.tumblr.com/post/37398560534/the-lambda-architecture-principles-for

[30] Careaga, J. (s. f.). Arquitectura Lambda vs Arquitectura Kappa ¿Cuál es el mejor enfoque para implementar un ambiente de trabajo para procesar big data? Recuperado de http://i2ds.org/wp-content/uploads/2018/02/arquitecturalambdavsarquitecturakappa.pdf

[31] Data processing architectures – Lambda and Kappa - Ericsson. (s. f.). Recuperado 20 de abril de 2019, de https://www.ericsson.com/en/blog/2015/11/data-processing-architectures--lambda-and-kappa

[32] From Lambda to Kappa: A Guide on Real-time Big Data Architectures. (s. f.). Recuperado 20 de abril de 2019, de https://www.talend.com/blog/2017/08/28/lambda-kappa-real-time-big-data-architectures/

[33] Zorrilla, M., & García-Saiz, D. (2017). Arquitecturas y tecnologías para el big data Este material se ofrece con licencia: Creative Commons BY-NC-SA 4.0. Recuperado de https://ocw.unican.es/pluginfile.php/2396/course/section/2473/tema 3.2 Arquitecturas y tecnologías para el big data.pdf

[34] Kappa Architecture - Where Every Thing Is A Stream. (s. f.). Recuperado 20 de abril de 2019, de http://milinda.pathirage.org/kappa-architecture.com/

[35] Questioning the Lambda Architecture - O'Reilly Media. (s. f.). Recuperado 20 de abril de 2019, de https://www.oreilly.com/ideas/questioning-the-lambda-architecture

[36] HDFS Architecture Guide. (s. f.). Recuperado 6 de mayo de 2019, de https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html

[37] General Parallel File System - Wikipedia, la enciclopedia libre. (s. f.). Recuperado 6 de mayo de 2019, de https://es.wikipedia.org/wiki/General_Parallel_File_System

[38] Google File System - Wikipedia, la enciclopedia libre. (s. f.). Recuperado 6 de mayo de 2019, de https://es.wikipedia.org/wiki/Google_File_System

[39] Soltis, S. R., Erickson, G. M., Preslan, K. W., & Soltis, S. R. (1997). The Global File System: A File System for Shared Disk Storage. *IEEE Transactions on Parallel and Distributed Systems*. Recuperado de http://www.diku.dk/undervisning/2003e/314/papers/soltis97global.pdf

[40] OLAP - Wikipedia, la enciclopedia libre. (s. f.). Recuperado 6 de mayo de 2019, de https://es.wikipedia.org/wiki/OLAP

[41] Hewitt, E., & Hewitt, E. (2011). *Cassandra : the definitive guide* (1st ed.). Recuperado de http://oreilly.com/catalog/0636920010852

[42] Teorema CAP - Wikipedia, la enciclopedia libre. (s. f.). Recuperado 10 de mayo de 2019, de https://es.wikipedia.org/wiki/Teorema_CAP

[43] Apache Spark™ - Unified Analytics Engine for Big Data. (s. f.). Recuperado 10 de mayo de 2019, de https://spark.apache.org/

[44] What is Apache MapReduce? | IBM. (s. f.). Recuperado 11 de mayo de 2019, de https://www.ibm.com/analytics/hadoop/mapreduce

[45] MapReduce 101: What It Is & How to Get Started - Talend. (s. f.). Recuperado 11 de mayo de 2019, de https://www.talend.com/resources/what-is-mapreduce/

[46] Apache Spark VS Hadoop Map Reduce | OpenWebinars. (s. f.). Recuperado 11 de mayo de 2019, de https://openwebinars.net/blog/apache-spark-vs-hadoop-map-reduce/

[47] What is the Difference Between Hadoop and Spark? | DIMENSIONLESS TECHNOLOGIES PVT.LTD. (s. f.). Recuperado 11 de mayo de 2019, de https://dimensionless.in/what-is-the-difference-between-hadoop-and-spark/

[48] An Introduction to and Evaluation of Apache Spark for Big Data Architectures. (s. f.). Recuperado 11 de mayo de 2019, de https://www.gartner.com/en/documents/3887566

[49] Apache Spark™ - What is Spark. (s. f.). Recuperado 11 de mayo de 2019, de https://databricks.com/spark/about

[50] Talend & Apache Spark: A Technical Primer and Overview. (s. f.). Recuperado 11 de mayo de 2019, de https://www.talend.com/blog/2017/09/15/talend-apache-spark-technical-primer/

[51] MapReduce vs Apache Spark- 20 Useful Comparisons To Learn. (s. f.). Recuperado 11 de mayo de 2019, de https://www.educba.com/mapreduce-vs-apache-spark/

[52] Druid: A Real-time Analytical Data Store. (s. f.). Recuperado 12 de mayo de 2019, de http://hexianghu.com/big-data/2015/03/15/druid/

[53] Apache Kudu - Fast Analytics on Fast Data. (s. f.). Recuperado 12 de mayo de 2019, de https://kudu.apache.org/

[54] Druid | Apache Druid (incubating) vs Kudu. (s. f.). Recuperado 12 de mayo de 2019, de http://druid.io/docs/latest/comparisons/druid-vs-kudu.html

[55] Apache Kudu in 5 Minutes – Lewis Gavin – Medium. (s. f.). Recuperado 12 de mayo de 2019, de https://medium.com/@lewisdgavin/apache-kudu-in-5-minutes-91e9371a4f8f

[56] Gupta, S., & Saxena, S. (2016). *Real-time big data analytics : design, process, and analyze large sets of complex data in real time*. Recuperado de https://www.worldcat.org/title/real-time-big-data-analytics-design-process-and-analyze-large-sets-of-complex-data-in-real-time/oclc/949230384&referer=brief_results

[57] Apache HBase – Apache HBase™ Home. (s. f.). Recuperado 18 de mayo de 2019, de https://hbase.apache.org/

[58] Kafka Connect HDFS — Confluent Platform. (s. f.). Recuperado 18 de mayo de 2019, de https://docs.confluent.io/current/connect/kafka-connect-hdfs/index.html

[59] Apache Spark and Hadoop HDFS: Working Together. (s. f.). Recuperado 18 de mayo de 2019, de https://databricks.com/blog/2014/01/21/spark-and-hadoop.html

[60] Running Spark on YARN - Spark 2.4.3 Documentation. (s. f.). Recuperado 18 de mayo de 2019, de https://spark.apache.org/docs/latest/running-on-yarn.html

[61] Introducing Apache Spark Data Sources API V2 - IBM Code. (s. f.). Recuperado 19 de mayo de 2019, de https://developer.ibm.com/code/2018/04/16/introducing-apache-spark-data-sources-api-v2/

[62] Apache Kudu - Developing Applications With Apache Kudu. (s. f.). Recuperado 19 de mayo de 2019, de https://kudu.apache.org/docs/developing.html

[63] DBMS | OLAP Operations - GeeksforGeeks. (s. f.). Recuperado 19 de mayo de 2019, de https://www.geeksforgeeks.org/dbms-olap-operations/

[64] Apache Spark Core Programming. (s. f.). Recuperado 19 de mayo de 2019, de https://www.tutorialspoint.com/apache_spark/apache_spark_core_programming.htm

[65] File System vs. Database - DZone Database. (s. f.). Recuperado 25 de mayo de 2019, de https://dzone.com/articles/which-is-better-saving-files-in-database-or-in-fil

[66] Understanding and Selecting Data Masking Solutions: Creating Secure and Useful Data. (2012). Recuperado de https://securosis.com/assets/library/reports/UnderstandingMasking_FinalMaster_V3.pdf

[67] Difference between Big Data Hadoop and Traditional RDBMS. (s. f.). Recuperado 22 de junio de 2019, de https://www.w3trainingschool.com/difference-big-data-hadoop-traditional-rdbms